Exhibit 4

```
<!--#include virtual="/common/properties/stdheader.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","I/O Points");%>
  <!--#include virtual="/common/properties/equipintro.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"points","I/O Points");%>
  <!--#include virtual="/common/properties/microblocks/mb_text_expandablebegin.jsp"-->
  <DIV style="height:10px; line-height:10px;"> </DIV>

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","I/O Points");%>
  <!--#include virtual="/common/properties/microblocks/mb_text_tablebegin.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"CTRL","Property");%>
  <!--#include virtual="/common/properties/microblocks/mb_bai.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M006","Property");%>
  <!--#include virtual="/common/properties/microblocks/mb_bao.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M007","Property");%>
  <!--#include virtual="/common/properties/microblocks/mb_bao.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","I/O Points");%>
  <!--#include virtual="/common/properties/microblocks/mb_text_tableend.jsp"-->

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","I/O Points");%>
  <!--#include virtual="/common/properties/microblocks/mb_text_expandableend.jsp"-->

  <BR>

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"control","Control Loops");%>
  <!--#include virtual="/common/properties/microblocks/mb_text_expandablebegin.jsp"-->
  <DIV style="height:10px; line-height:10px;"> </DIV>

  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M004","Direct Acting Loop");%>
  <!--#include virtual="/common/properties/microblocks/mb_pid2da.jsp"-->
  <BR>
  <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M005","Reverse Acting Loop");%>
  <!--#include virtual="/common/properties/microblocks/mb_pid2ra.jsp"-->
```

```
    40 <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","I/O Points");%>
       <!--#include virtual="/common/properties/microblocks/mb_text_expandableend.jsp"-->

       <BR>

    45 <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M012","Setpoints");%>
       <!--#include virtual="/common/properties/microblocks/mb_text_sep.jsp"-->


       <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M002","Property");%>
    50 <!--#include virtual="/common/properties/microblocks/mb_setpt.jsp"-->

       <BR> 
       <BR>
       <% cjMBPropertyPage = new
com.controlj.green.web.jsp.controls.MicroBlockString(loc.getBinder(),"M014","Temperature Colors");%>
    55 <!--#include virtual="/common/properties/microblocks/mb_tstclr.jsp"-->

       <!--#include virtual="/common/properties/stdfooter.jsp"-->
```

Exhibit 5

```
<HTML>
<HEAD>
<LINK ID="ctrlStyles" REL=STYLESHEET TYPE="text/css"
HREF="/common/pagestyles/standard/controlstyles.css">
<%@import="com.controlj.green.web.jsp.controls.*, com.controlj.green.core.data.CoreNode"%>
<% Location loc = new Location(request); %>
<% String cjMicroblock; %>
<% String cjMBString; %>
<% MicroBlockString cjMBPropertyPage; %>

<% boolean cjInTable = false; %>

<SCRIPT LANGUAGE="JavaScript" src="/common/debug/cjdebugpub.js"></SCRIPT>

<SCRIPT LANGUAGE="JavaScript" src="/common/util/CJElementHelper.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/util/BrowserApi.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/exception.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/CJEvent.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/primitive.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBase.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/stylemgr.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBase.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/TextValidation.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/WidgetTextInput.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/iconhelper.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/WidgetImageButton.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/WidgetBandsBar.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBase.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlImageButton.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlTextInput.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/WidgetDropList.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlDropList.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlDateInput.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlTimeInput.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlSlider.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBacnetOID.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlColorMap.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlSetPoint.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlImageMap.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/WidgetImageAreaLabel.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlColorMap.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlMinSecInput.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/common/util/classhelper.js"></SCRIPT>

<!--#include virtual="/common/util/StdActionPage.jsp" -->


</HEAD>

<BODY onLoad="onPageLoaded()" onUnload="CJElementHelper.flushList()"
```

```
style="background-image:url(/common/pagestyles/standard/GreyText
    50 <DIV ID="cjOuter" style="padding:10px;">

       <IFRAME NAME="primitivesubmit" ID="primitivesubmit"  WIDTH="0" HEIGHT="0"
SRC="/common/darkblank.html" style="visibility:hidden;"></I
       <IFRAME NAME="primitiveupdate" ID="primitiveupdate"  WIDTH="0" HEIGHT="0"
SRC="/common/darkblank.html" style="visibility:hidden;"></I
       <IFRAME NAME="jumprequest" ID="jumprequest"  WIDTH="0" HEIGHT="0" SRC="/common/darkblank.html"
style="visibility:hidden;"></IFRAME>
    55
```

Exhibit 6

```
</DIV>
<SCRIPT language="JavaScript">
   getUpdate(true);     // Get update and force field refresh
   top.setActBusy(false);
</SCRIPT>
</BODY>
</HTML>
```

Exhibit 7

```
<SCRIPT LANGUAGE="JavaScript">
/*****************************************************************************
                        AUTOMATED LOGIC CORPORATION
                Copyright (c) 1999 - 2000 All Rights Reserved
        This document contains confidential/proprietary information.
 *****************************************************************************

 @(#)StdActionPage.jsp
 Initial Author: Steve Appling

 Description: Contains various functions that commonly used on action
 pages.

 $Log:  $
*****************************************************************************/

var AdvPropertiesRule;

function onPageLoaded()
{
   if (window.parent && window.parent.mainLoaded)
   {
      var cjOuter = document.getElementById("cjOuter");
      CJAssert(cjOuter, "cjOuter container not found");
      window.parent.actionFrameLoaded(window, cjOuter);
   }
   PrimitiveBase.addUIListener(this);

   var ruleset = document.styleSheets["ctrlStyles"].rules;
   if (ruleset)
   {
      AdvPropertiesRule = ruleset[0].style;
   }
}

var _cjPageRefresh = 4000;
var _cjUpdateCount = 0;
var _cjGetValuesFromField=false;
function updateFinished()
{
//window.status=_cjUpdateCount++;
   setTimeout("getUpdate(_cjGetValuesFromField)", _cjPageRefresh);
}


function getUpdate(refreshField)
{
   window.parent.setActBusy(true);
```

```
        document.getElementById("primitiveupdate").src="/common/servlet/primitiveupdate?wbs="+<%=
loc.getWBS().getId() %>+"&getFieldValues

    }

55

    function valueChangedByUser(from)
    {
        window.parent.currentServerPage = this;
60      window.parent.setActionButtonState(false);
    }

    function postChanges()
    {
65      var changes = PrimitiveBase.changedPrimitives();
        var commandList = "/common/servlet/primitivesubmit?wbs="+<%= loc.getWBS().getId() %>;

        for (var i=0; i<changes.length; i++)
        {
70          var newChange = changes[i].getId()+"="+escape(changes[i].toString());
            commandList += "&"+newChange;
        }
        //alert(commandList);
        window.parent.setActBusy(true);
75      document.all.primitivesubmit.src = commandList;
    }

    function cancelChanges()
    {
80      PrimitiveBase.resetAllChangedPrimitives();
    }

    function jumpToTreeLocation(treeVarName, location, page)
    {
85      window.parent.setNavBusy(true);
        var jumpRequest = "/common/servlet/treejumprequest" +
                          "?wbs=<%= loc.getWBS().getId() %>" +
                          "&arehere=<%= loc %>"+
                          "&treename="+ treeVarName +
90                        "&path="+ location +
                          (page!=null?"&page="+page:"");
        //alert(jumpRequest);
        var jumpframe = document.getElementById("jumprequest");
        if (jumpframe)
95      {
            document.all.jumprequest.src = jumpRequest;
        }
        else
        {
```

```
100         alert("Error: Action page contains no jumprequest IFRAME");
        }
    }
    var disableControls = new Object();
105
    </SCRIPT>
```

Exhibit 8

```
/*****************************************************************************
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
        This document contains confidential/proprietary information.
*****************************************************************************

@(#)primitive.js
Initial Author: Doyle Davidson

Description: Standard primitives

$Log: /green/webroot/common/controls/Primitive.js $
*
* 39    12/14/99 11:41a Ddavidson
* Fixed exception message in addUIListener
*
* 38    12/13/99 3:41p Ddavidson
* added isPageLocal().
* fixed table setColumn to do notifyListeners();
* made notifyListeners pass the "from" as a second param
*
* 37    12/08/99 1:24p Ddavidson
* Modified PrimitiveEnum has handle any "key" value. Doesn't have to be
* just numbers, can also be any string.
*
* 36    12/08/99 10:10a Jomisore
* Fix for getting number of rows when there is no data.
*
* 35    12/06/99 11:25a Ddavidson
* added code to notify listeners when all primitives are reset WHEN THE
* PRIMITIVE DIDN'T CHANGE.
*
* 34    12/03/99 5:59p Jomisore
* Fixed the string format for tables.
*
* 33    11/24/99 1:45p Jomisore
* Fixed merge conflicts.
*
* 32    11/24/99 1:39p Jomisore
* Table primitive 1st version.
*
* 31    11/18/99 12:57p Tirish
*
* 30    11/17/99 8:39a Tirish
*
* 29    11/16/99 9:51a Ddavidson
```

```
    * Made clearChanged() an instance method.  This allows a primitive to
    * override it.
    * PrimitiveBase.clearChangedFlag() now invokes instance method.
50  *
    * 28    11/09/99 8:49a Ddavidson
    * fixed PrimitiveBitString_setBit to invoke underlying setValue() call in
    * order to trigger listeners.
    *
55  * 27    11/05/99 9:56a Sappling
    * Added PrimTable basic support
    *
    * 26    11/04/99 10:14a Ddavidson
    * added "changed=false" to resetValue() method.
60  *
    * 25    11/03/99 9:42a Jomisore
    * Changed it so that findPrimitive throws an exception again when the
    * primitive is not found.
    *
65  * 24    11/01/99 4:12p Ddavidson
    * Fixed notifyUIListener to pass the primitive being changed in the
    * callback
    *
    * 23    10/07/99 3:14p Sappling
70  * Removed exception when primitive not found
    *
    * 22    9/20/99 3:45p Wdover
    * Added clearChangedFlag static method to PrimitiveBase
    *
75  * 21    9/15/99 5:11p Wdover
    * Modified to disable cancel on local primitives
    *
    * 20    9/15/99 5:04p Wdover
    * Modified to disable accept/cancel on primitives not connected to core
80  *
    * 19    9/14/99 10:39a Wdover
    * Modified to use negative max value for null minimum number values
    *
    * 18    9/10/99 4:43p Jomisore
85  * Fixed a problem where when setting the "minimum" restriction it was
    * checking against "this.maximum" which wasn't already set.
    *
    * 17    9/09/99 3:36p Wdover
    * Modified to enable bad value checking
90  *
    * 16    9/09/99 2:20p Jomisore
    * Fixed a problem with checking precision restrictions of numbers.
```

```
     *
     * 15    9/09/99 11:06a Sappling
 95  * Disabled setMasterValue after setValue has been called
     *
     * 14    9/09/99 9:48a Jomisore
     * Fixed a bug with the "getMinimum" method in the Number primitive.
     *
100  * 13    9/08/99 4:26p Wdover
     * Adding function to PrimitiveBase to notify pages when user changes
     * values (for accept/cancel); Changed PrimitiveBacnetOID to use -1 for
     * undefined values
     *
105  * 12    9/03/99 3:33p Tirish
     * changed the restrictions to accept null values
     *
     * 10    9/03/99 11:28a Tirish
     * Restrictions added and constructors changed
110  *
     * 9     8/12/99 4:37p Tirish
     * added a toString method for Core. Constructor can now accept both a
     * string representation of the primitive and a value for it's initial
     * value.
115  * primitiveBitString is now stored and returned as a  string
     *
     * 8     8/04/99 2:54p Wdover
     * Change signed bit shifts to unsigned bit shifts
     *
120  * 7     7/23/99 9:52a Ddavidson
     * added some CJAssert calls - improved some code readability.
     *
     * 6     7/21/99 2:16p Ddavidson
     * made all 'set' method include a 'from' parameter
125  *
     * 5     7/20/99 5:07p Ddavidson
     * added PrimitiveExpr class that is an evaluated primitive.
     *
     * 4     7/19/99 4:20p Ddavidson
130  * removed used of 'byte' keyword
     *
     * 3     7/19/99 10:04a Ddavidson
     * Fixed exception messages to use double quotes instead of braces.
     *
135  * 2     7/16/99 1:55p Ddavidson
     * code review fixes
     *
     * 1     7/14/99 2:40p Ddavidson
```

```
 * JavaScript primitive objects
140
*****************************************************************************/

/*****************************************************************************
 *****************************************************************************
145
  CLASS: PrimitiveBase

  ATTRIBUTES:
    id          String   A unique string identifying the primitive instance
150

    listeners   Array    An array of object supporting the "valueChanged"
                         function that are notified when the value changes.
    ui listeners Array   An array of objects supporting the "valueChangedByUser"
155                      function that are notified when the value changes
                         as a result of user actions.
    data        object   The current valid value of the primitive
    origData    object   The original/last value specified from CORE
    changed     boolean  True if the value has been set using setValue
160                        note that this will disable setMasterValue
*****************************************************************************
*****************************************************************************/

function PrimitiveBase( id, writable, initData )
165 {
   if ( id == null )  // handle case of assignment to prototype
      return;

   this.id   = id;
170   this.writable = writable;
   this.listeners = new Array;
   // if the initData is a string or a String object
   // convert it to it's internal representation
   if(initData!=null && typeof initData.valueOf() == "string")
175   {
      initData=this.fromString(initData);
   }
   this.origData = initData;

180   this.resetData();

   if ( PrimitiveBase.allPrimitives[id] != null )
   {
      throw new CJException.DuplicateItem(
```

```
185           "Primitive named \"" + id + "\" already exists" );
      }
      PrimitiveBase.allPrimitives[id] = this;
      this.changed = false;
      this.badValue = false;
190   this.pageLocal = false;
   }

   /*************************************************************************
     Member accessors:
195 *************************************************************************/

   // return primitive's ID
   function _PrimitiveBase_getId()
   {
200   return this.id;
   }

   // return true if the primitive is different from it's original value
   function _PrimitiveBase_isChanged()
205 {
      return (this.changed);
   }

   // return true if primitive is undefined (value is null)
210 function _PrimitiveBase_isUndefined()
   {
      return ( this.data == null );
   }

215 // return true if primitive is pageLocal
   function _PrimitiveBase_isPageLocal()
   {
      return ( this.pageLocal );
   }
220
   // return true if one can write to the primitive
   function _PrimitiveBase_isWritable()
   {
      return this.writable;
225 }

   // return true if value from primitive is bad
   function _PrimitiveBase_hasBadValue()
   {
230   return this.badValue;
```

```
    }

    // return the primitive's value
    function _PrimitiveBase_getValue()
235 {
       return this.data;
    }

    // clear the "changed" flag
240 function _PrimitiveBase_clearChanged( )
    {
       this.changed = false;
    }

245 // return the primitive's value as a string
    // this is the default method
    function _PrimitiveBase_toString()
    {
       //There is a little bit of a debate over what this should return
250    //when the value is null. For now we are returning the empty string "".
       if ( this.data == null )
          return "";
       else
          return this.data.toString();
255 }
    // converts the string given into the proper format
    //This default method does nothing
    function _PrimitiveBase_fromString(text)
    {
260       return text;
    }
    // checks if the value is valid
    //This default method will accept any value
    //and is here only as a placeholder
265 function _PrimitiveBase_validateValue( test )
    {

    }

270 /**************************************************************************
     Class : PrimitiveBase
     Method: notifyListeners
     Params: Listener except
     Return: none
275  Description: Walks the list of listeners and invokes the
       "valueChanged() method for each one.  If an 'except' listener is
```

```
   specified, then that listener, if present, will be skipped and
   therefore not notified.
*************************************************************************/
function _PrimitiveBase_notifyListeners( except )
{
   for ( var i=0; i<this.listeners.length; i++ )
   {
      if ( this.listeners[i] != null && except != this.listeners[i] )
      {
         this.listeners[i].valueChanged( this, except );
      }
   }
}

/*************************************************************************
 Class : PrimitiveBase
 Method: notifyUIListeners
 Params: Primitive prim    The primitive being changed.
         Object    from    The object causing the change
 Return: none
 Description: Walks the list of uilisteners and invokes the
   "valueChangedByUser() method for each one.
*************************************************************************/
function _PrimitiveBase_notifyUIListeners( prim, from )
{
   for ( var i=0; i < PrimitiveBase.uilisteners.length; i++ )
   {
      if ( PrimitiveBase.uilisteners[i] != null )
      {
         PrimitiveBase.uilisteners[i].valueChangedByUser( prim, from );
      }
   }
}

/*************************************************************************
 Class : PrimitiveBase
 Method: setMasterValue
 Params: object newValue
 Return: none
 Description: Sets the master value of the primitive unconditionally
   and the current value only if never set by the user.
   It then notifies all listeners of the change.

  Essentially don't set the current value the first time in if the
  user has already set a current value.  Meaning he has changed the
  value before the master value was initially loaded.
```

```
   ************************************************************************/
function _PrimitiveBase_setMasterValue( newData )
{
   // If master has been set before, or current was never set and
   // the user has not changed the value, then set the current value.
   newData = this.fromString(newData);

   this.origData = newData;

   if (( this.origData != null || this.data == null ) && !this.changed)
   {
      this.resetData();
      this.changed = false;
      this.notifyListeners();
   }
}

/************************************************************************
 Class : PrimitiveBase
 Method: setMasterUndefined
 Params:
 Return: none
 Description: Sets the master value of the primitive to the undefined state
   and the current value only if never set by the user.
   It then notifies all listeners of the change.

************************************************************************/
function _PrimitiveBase_setMasterUndefined()
{
   this.origData = null;

   if (!this.changed)
   {
      this.resetData();
      this.changed = false;
      this.notifyListeners();
   }
}


/************************************************************************
 Class : PrimitiveBase
 Method: setBadValue
 Params: boolean badValueFlag - new value of badValueflag
 Return: none
```

```
     Description: Sets the bad value flag of the primitive unconditionally
370    if different from its current value and if the user has not changed
       the primitive. It then notifies all listeners of the change so that
       they can act appropriately.

    ************************************************************************/
375 function _PrimitiveBase_setBadValue( badValueFlag )
    {
       if ((this.badValue != badValueFlag) && !this.changed)
       {
          this.badValue = badValueFlag;
380       this.notifyListeners();
       }
    }

    // Set the primitive's value - returns 'true' if successful.
385 /************************************************************************
     Class : PrimitiveBase
     Method: setValue
     Params: object newData,  Listener from
     Return: none
390  Description: Attempts to set the current value of the primitive.
       The primitive must be writable and the value must be valid.
       If either of these conditions fails, and exception will be thrown.
       Otherwise, the value is set and the listeners are notified.  The
       listener specified in the 'from' value, if any, will be excepted
395    from the notification.
     Exceptions: ReadOnly
    ************************************************************************/
    function _PrimitiveBase_setValue( newData, from )
    {
400    // if writable, set new value and call listeners
       if ( this.writable == false )
       {
          throw new CJException.ReadOnly(
             "Cannot set value for primitive \"" + this.id + "\"" );
405    }
       this.validateValue( newData );

       // update value
       this.data = newData;
410    this.changed = (this.data != this.origData);
       this.notifyListeners( from );
       if (!this.pageLocal)
       {
          PrimitiveBase.notifyUIListeners(this, from );
```

```
415     }
    }

    // Set the primitive's value - returns 'true' if actually changed.
    /**************************************************************************
420  Class : PrimitiveBase
     Method: resetValue
     Params: Listener from
     Return: Boolean - true if the value changed
     Description: Resets the current value to the master value if it is
425    different.  If a change occurs, the listeners will be notifier with
       the possible exception of the listener that reset the value.
    **************************************************************************/
    function _PrimitiveBase_resetValue(from)
    {
430    // if writable, set new value and call listeners
       if (this.writable && this.changed == true)
       {
          // update value
          this.resetData();
435       this.changed = false;
          this.notifyListeners( from );
          if (!this.pageLocal)
          {
             PrimitiveBase.notifyUIListeners(this, from);
440       }
          return true;
       }
       else
          return false;
445 }


    // add listener to the primitive.  It must have "valueChanged" function
    /**************************************************************************
450  Class : PrimitiveBase
     Method: addListener
     Params: Listener who
     Return: none
     Description: Adds the specified to the list of listeners if it isn't
455    already on the list.  A listened is defined as any object that has
       a "valueChanged" function.
    **************************************************************************/
    function _PrimitiveBase_addListener( who )
    {
460    CJAssert( who != null, "listened is null" );
```

```
      // don't add it object doesn't have a valueChanged method
      if ( who.valueChanged == null )
      {
         throw new CJException.InvalidClass(
465         "PrimitiveBase_addListener - " +
            "attempt to add listen with no valueChanged() method" );
      }

      // check for already added
470   var i = 0;
      for ( i=0; i<this.listeners.length; i++ )
      {
         if ( this.listeners[i] == who )
            return;
475   }

      // add to end
      this.listeners[this.listeners.length] = who;
   }
480
   /**************************************************************************
    Class : PrimitiveBase
    Method: removeListener
    Params: Listener who
485 Return: none
    Description: Removes the specified listens from the list of listeners.
   **************************************************************************/
   function _PrimitiveBase_removeListener( who )
   {
490   var i = 0;
      for ( i=0; i<this.listeners.length; i++ )
      {
         if ( this.listeners[i] == who )
            this.listeners[i] = null;
495   }
   }

   // add uilistener to the primitive.  It must have "valueChangedByUser" function
   /**************************************************************************
500 Class : PrimitiveBase
    Method: addUIListener
    Params: Listener who
    Return: none
    Description: Adds the specified to the list of listeners if it isn't
505   already on the list.  A listened is defined as any object that has
      a "valueChanged" function.
```

```
   ************************************************************************/
   function _PrimitiveBase_addUIListener( who )
   {
510   CJAssert( who != null, "listened is null" );
      // don't add it object doesn't have a valueChanged method
      if ( who.valueChangedByUser == null )
      {
         throw new CJException.InvalidClass(
515         "PrimitiveBase_addUIListener - " +
            "attempt to add listen with no valueChangedByUser() method" );
      }

      // check for already added
520   var i = 0;
      for ( i=0; i<PrimitiveBase.uilisteners.length; i++ )
      {
         if ( PrimitiveBase.uilisteners[i] == who )
            return;
525   }

      // add to end
      PrimitiveBase.uilisteners[PrimitiveBase.uilisteners.length] = who;
   }
530
   /************************************************************************
    Class : PrimitiveBase
    Method: removeUIListener
    Params: Listener who
535 Return: none
    Description: Removes the specified listens from the list of listeners.
   ************************************************************************/
   function _PrimitiveBase_removeUIListener( who )
   {
540   var i = 0;
      for ( i=0; i<PrimitiveBase.uilisteners.length; i++ )
      {
         if ( PrimitiveBase.uilisteners[i] == who )
            PrimitiveBase.uilisteners[i] = null;
545   }
   }

   /************************************************************************
    Class : PrimitiveBase
550 Method: findPrimitve (static class method)
    Params: String id
    Return: Primitive
```

```
     Description: Returns the primitive with the specified ID.
     Exceptions: NullValue, NotFound
555 ************************************************************************/
    function _PrimitiveBase_findPrimitive( id )
    {
       if ( id == null )
       {
560       throw new CJException.NullValue(
             "PrimitiveBase_findPrimitive - id is NULL" );
       }

       if ( PrimitiveBase.allPrimitives[id] == null )
565    {
          throw new CJException.NotFound(
             "PrimitiveBase_findPrimitive - primitive for id \""
             + id + "\" not found" );
       }
570
       return PrimitiveBase.allPrimitives[id];
    }

    /************************************************************************
575  Class : PrimitiveBase
     Method: changedPrimitives (static class method)
     Params: none
     Return: Array of primitives
     Description: Rreturns an array of all the primitives that have been
580    changed.  This is determined by the isChanged() method.
    ************************************************************************/
    function _PrimitiveBase_changedPrimitives( )
    {
       var changed = new Array;
585    for ( var id in PrimitiveBase.allPrimitives )
       {
          var prim = PrimitiveBase.allPrimitives[id];
          if ( prim.isChanged() )
             changed[changed.length] = prim;
590    }

       return changed
    }

595 /************************************************************************
     Class : PrimitiveBase
     Method: setPrimitiveValue (static class method)
     Params: String id, object newData
```

```
    Return: none
600 Description: Finds the specified primitive by ID and then invokes the
      setValue() method.
    Exceptions: NullValue, NotFound
   ************************************************************************/
   function _PrimitiveBase_setPrimitiveValue( id, newData, from )
605 {
      // findPrimitive will throw exception if ID is bad
      PrimitiveBase.findPrimitive( id ).setValue( newData, from );
   }

610 /************************************************************************
    Class : PrimitiveBase
    Method: setPrimitiveBadValue (static class method)
    Params: String id, boolean badValueFlag
    Return: none
615 Description: Finds the specified primitive by ID and then invokes the
      setBadValue() method.
    Exceptions: NullValue, NotFound
   ************************************************************************/
   function _PrimitiveBase_setPrimitiveBadValue( id, badValueFlag )
620 {
      // findPrimitive will throw exception if ID is bad
      PrimitiveBase.findPrimitive( id ).setBadValue( newData, from );
   }

625 /************************************************************************
    Class : PrimitiveBase
    Method: resetAllChangedPrimitives (static class method)
    Params: from - object that called
    Return: none
630 Description: Resets all changed primitives to the original values
    Exceptions:
   ************************************************************************/
   function _PrimitiveBase_resetAllChangedPrimitives(from)
   {
635   for ( var id in PrimitiveBase.allPrimitives )
      {
         var prim = PrimitiveBase.allPrimitives[id];
         if ( prim.isChanged() && (!prim.pageLocal))
            prim.resetValue(from);
640      else
         {
           // And notify all listeners even if nothing changes.
           // This should reset any invalid controls too.
           prim.notifyListeners( null );
```

```
645       }
      }
   }

   /*************************************************************************
650  Class : PrimitiveBase
    Method: setPrimitiveMasterValue (static class method)
    Params: String id, object newData
    Return: none
    Description: Finds the specified primitive by ID and then invokes the
655   setMasterValue() method.
    Exceptions: NullValue, NotFound

    NOTE: THIS METHOD SHOULD ONLY BE INVOKED INDIRECTLY FROM THE CORE VIA
    A SERVLET/JSP PAGE!!!!!!
660 *************************************************************************/
   function _PrimitiveBase_setPrimitiveMasterValue( id, newData )
   {
      // findPrimitive will throw exception if ID is bad
      PrimitiveBase.findPrimitive( id ).setMasterValue( newData );
665 }

   /*************************************************************************
    Class : PrimitiveBase
    Method: clearChangedFlag (static class method)
670  Params: String id
    Return: none
    Description: Finds the specified primitive by ID and then clears its
      changed flag. This signifies that core has received the update from
      the user.
675  Exceptions: NullValue, NotFound

    NOTE: THIS METHOD SHOULD ONLY BE INVOKED INDIRECTLY FROM THE CORE VIA
    A SERVLET/JSP PAGE!!!!!!
   *************************************************************************/
680 function _PrimitiveBase_clearChangedFlag( id )
   {
      // findPrimitive will throw exception if ID is bad
      PrimitiveBase.findPrimitive( id ).clearChanged();
   }
685
   /*************************************************************************
    Class : PrimitiveBase
    Method: helper_to8HexString
    Params: none
690  Return: 8 char hex string.
```

```
     Description: Converts the current data value to a 8 char (4 byte)
       hex string with leading zeros.
    ************************************************************************/
    function _PrimitiveBase_helper_to8HexString( )
695 {
       if ( this.data == null )
       {
          return "00000000";
       }
700
       var rval = "00000000" + this.data.toString(16);
       return rval.substr( rval.length-8, 8 );
    }

705 /************************************************************************
     Class : PrimitiveBase
     Method: setPageLocal
     Params: state
     Return: none
710  Description: accessor to set the pageLocal state.  When set to true,
                  the primitive will not call UIListeners when it changes value.
    ************************************************************************/
    function _PrimitiveBase_setPageLocal(state)
    {
715    this.pageLocal = state;
    }


    /************************************************************************
720  Class : PrimitiveBase
     Method: resetData
     Params:
     Return: none
     Description: Sets this.data to the master value.
725 ************************************************************************/

    function _PrimitiveBase_resetData()
    {
       this.data = this.origData;
730 }



    PrimitiveBase.prototype.PrimitiveBase   = PrimitiveBase;
735 PrimitiveBase.prototype.toString        = _PrimitiveBase_toString;
    PrimitiveBase.prototype.fromString      = _PrimitiveBase_fromString;
```

```
    PrimitiveBase.prototype.getId           = _PrimitiveBase_getId;
    PrimitiveBase.prototype.isChanged       = _PrimitiveBase_isChanged;
    PrimitiveBase.prototype.isWritable      = _PrimitiveBase_isWritable;
740 PrimitiveBase.prototype.isUndefined     = _PrimitiveBase_isUndefined;
    PrimitiveBase.prototype.isPageLocal     = _PrimitiveBase_isPageLocal;
    PrimitiveBase.prototype.clearChanged    = _PrimitiveBase_clearChanged;
    PrimitiveBase.prototype.validateValue   = _PrimitiveBase_validateValue;
    PrimitiveBase.prototype.getValue        = _PrimitiveBase_getValue;
745 PrimitiveBase.prototype.setValue        = _PrimitiveBase_setValue;
    PrimitiveBase.prototype.setMasterValue  = _PrimitiveBase_setMasterValue;
    PrimitiveBase.prototype.setMasterUndefined =  _PrimitiveBase_setMasterUndefined;
    PrimitiveBase.prototype.hasBadValue     = _PrimitiveBase_hasBadValue;
    PrimitiveBase.prototype.setBadValue     = _PrimitiveBase_setBadValue;
750 PrimitiveBase.prototype.resetValue      = _PrimitiveBase_resetValue;
    PrimitiveBase.prototype.addListener     = _PrimitiveBase_addListener;
    PrimitiveBase.prototype.removeListener  = _PrimitiveBase_removeListener;
    PrimitiveBase.prototype.notifyListeners = _PrimitiveBase_notifyListeners;
    PrimitiveBase.prototype.setPageLocal    = _PrimitiveBase_setPageLocal;
755 PrimitiveBase.prototype.resetData       = _PrimitiveBase_resetData;

    PrimitiveBase.findPrimitive             = _PrimitiveBase_findPrimitive;
    PrimitiveBase.changedPrimitives         = _PrimitiveBase_changedPrimitives;
    PrimitiveBase.setPrimitiveValue         = _PrimitiveBase_setPrimitiveValue;
760 PrimitiveBase.setPrimitiveBadValue      = _PrimitiveBase_setPrimitiveBadValue;
    PrimitiveBase.setPrimitiveMasterValue   = _PrimitiveBase_setPrimitiveMasterValue;
    PrimitiveBase.notifyUIListeners         = _PrimitiveBase_notifyUIListeners;
    PrimitiveBase.addUIListener             = _PrimitiveBase_addUIListener;
    PrimitiveBase.removeUIListener          = _PrimitiveBase_removeUIListener;
765 PrimitiveBase.resetAllChangedPrimitives = _PrimitiveBase_resetAllChangedPrimitives;
    PrimitiveBase.clearChangedFlag          = _PrimitiveBase_clearChangedFlag;

    PrimitiveBase.allPrimitives = new Array;
    PrimitiveBase.uilisteners = new Array;
770 /*****************************************************************************
     *****************************************************************************
                              END PrimitveBase CLASS
     *****************************************************************************
     *****************************************************************************/
775



780

    /*****************************************************************************
```

```
 ***************************************************************************

785   CLASS: PrimitiveString

      The string class is a simple primitive whos data type is a String

       Restrictions:
790       maxlength - max string length (null = no limit)

    ***************************************************************************
    ***************************************************************************/
    function PrimitiveString( id, writable, initData, maxlength )
795 {
       this.setMaxLength(maxlength);
       this.PrimitiveBase( id, writable, initData );
    }

800 /***************************************************************************
     Class : PrimitiveString
     Method: getMaxLength, hasMaxLength, setMaxLength and validateValue
     Params: varies
     Return: varies
805  Description: The restriction for the PrimitiveString is its length
                  these methods handle this restriction.
    ***************************************************************************/
    function _PrimitiveString_getMaxLength()
    {
810    if ( this.maxlength == null )
          return Number.MAX_VALUE;
       else
          return this.maxlength;
    }
815
    function _PrimitiveString_hasMaxLength()
    {
       return this.maxlength != null;
    }
820
    function _PrimitiveString_setMaxLength(maxlen)
    {
       if ( maxlen < 0 )
       {
825       throw new CJException.InvalidValue(
             "max length " + maxlen + " must be >= 0" );
       }
```

```
      this.maxlength=maxlen;
}

/**************************************************************************
 Class : PrimitiveString
 Method: validateValue
 Params: String test
 Return: null
 Description: checks the test value against the primitive's restrictions
**************************************************************************/
function _PrimitiveString_validateValue( test )
{
   //check the test value's length.
   if ( this.hasMaxLength() )
   {
      if( test.length > this.getMaxLength() )
         throw new CJException.InvalidValue("String of length " + test.length +
                                    " exceeds limit of " + this.getMaxLength());
   }
}

PrimitiveString.prototype = new PrimitiveBase();
PrimitiveString.prototype.validateValue= _PrimitiveString_validateValue;
PrimitiveString.prototype.maxlength    = null;
PrimitiveString.prototype.getMaxLength = _PrimitiveString_getMaxLength;
PrimitiveString.prototype.hasMaxLength = _PrimitiveString_hasMaxLength;
PrimitiveString.prototype.setMaxLength = _PrimitiveString_setMaxLength;




/**************************************************************************
 **************************************************************************

  CLASS: PrimitiveBoolean

  The boolean class is a simple primitive whoses data type is a Boolean

   Restrictions:
      none

**************************************************************************
**************************************************************************/
function PrimitiveBoolean( id, writable, initData )
```

```
875 {
       this.PrimitiveBase( id, writable, initData );
    }

    /*************************************************************************
880  Class : PrimitiveBoolean
     Method: fromString
     Params: String text
     Return: Boolean
     Description: converts the string given into a boolean
885 *************************************************************************/
    function _PrimitiveBoolean_fromString(text)
    {
       switch(text)
       {
890       //returns a null if given a null
          case null: return null;
          //returns proper value when correct string is passed
          case "true": return true;
          case "false": return false;
895       //throws an exception if an invalid string is passed
          default: throw new CJException.InvalidValue("\nPrimitiveBoolean: fromString() must be given "+
                                                      "\n\"true\" or \"false\". It received \""+text+"\".");
       }
    }
900
    /*************************************************************************
     Class : PrimitiveBoolean
     Method: validateValue
     Params: Boolean test
905  Return: null
     Description: checks the test value against the primitive's restrictions
    *************************************************************************/
    function _PrimitiveBoolean_validateValue( test )
    {
910    //check if the test value is not one of the valid values.
       if ( test!=null && test!=true && test!="true" && test!="false" && test!=false )
       {
          throw new CJException.InvalidValue("Boolean must be true, false or null recieved \"" + test + "\".");

       }
915 }
    PrimitiveBoolean.prototype = new PrimitiveBase();
    PrimitiveBoolean.prototype.fromString = _PrimitiveBoolean_fromString;
    PrimitiveBoolean.prototype.validateValue = _PrimitiveBoolean_validateValue;
```

```
920

    /**************************************************************************
     **************************************************************************
925
      CLASS: PrimitiveNumber

      The number class is a simple primitive whoses data type is a Number

930    Restrictions:
          minimum        - the minimum allowed value (null=no limit)
          maximum        - the maximum allowed value (null=no limit)
          resnumerator   - the resolution, expressed in fractional form (null=none)
                         - if the denominator is null then this is the integer resolution
935       resdenominator - if both resnumerator and resdenominator are set then use
                           a fractional resolution
          tolerance      - the tolerated margin of error.  Value must be within
                           this tolerance of the fractional resolution.  For example,
                           1.33 must have a tolerance of 0.01 if res = 1.0/3.0
940                        The default is 0.0, must be an exact multiple, which means
                           that 1/3 would never be tolerated for non-integral values.
          precision      - how many significant digits are allowed (null=no limit)


945 **************************************************************************
    **************************************************************************/
    function PrimitiveNumber( id, writable, initData, minimum, maximum, resnumerator, resdenominator,
tolerance, precision)
    {
       this.setMaximum(maximum);
950    this.setMinimum(minimum);
       this.setResNumerator(resnumerator);
       this.setResDenominator(resdenominator);
       this.setTolerance(tolerance);
       this.setPrecision(precision);
955    this.PrimitiveBase( id, writable, initData );
    }

    /**************************************************************************
     Class : PrimitiveNumber
960  Method: fromString
     Params: String text
     Return: Number
     Description: converts the string given into a number
    **************************************************************************/
```

```
function _PrimitiveNumber_fromString(text)
{
   if(text==null){ return null;}
   //parse out the first number in the string.
   //must begin with a number.
   var number=parseFloat(text);
   //if a number was found return it
   if(!isNaN(number)){ return number;}
   //throw an exception if the conditions above were not met
   throw new CJException.InvalidValue("\nPrimitiveNumber: fromString() must be given a string "+
                                      "\nbeginning with a number.It received \""+text+"\".");
}

/*********************************************************************
 Class : PrimitiveNumber
 Method: getMaximum, hasMaximum and setMaximum
 Params: varies
 Return: varies
 Description: One restriction for the PrimitiveNumber is its maximum
              allowed value these methods handle this restriction.
*********************************************************************/
function _PrimitiveNumber_getMaximum()
{
   if ( this.maximum == null )
      return Number.MAX_VALUE;
   else
      return this.maximum;
}

function _PrimitiveNumber_hasMaximum()
{
   return this.maximum != null;
}

function _PrimitiveNumber_setMaximum(max)
{
   if ( max < this.minimum )
   {
      throw new CJException.InvalidValue(
         "maximum value " + max + " must be >= the minimum value of "+this.minimum+"." );
   }
   this.maximum=max;
}
/*********************************************************************
 Class : PrimitiveNumber
 Method: getMinimum, hasMinimum and setMinimum
```

```
     Params: varies
     Return: varies
     Description: One restriction for the PrimitiveNumber is its maximum
                  allowed value these methods handle this restriction.
1015 ************************************************************************/
     function _PrimitiveNumber_getMinimum()
     {
        if ( this.minimum == null )
           return -Number.MAX_VALUE;
1020    else
           return this.minimum;
     }

     function _PrimitiveNumber_hasMinimum()
1025 {
        return this.minimum != null;
     }

     function _PrimitiveNumber_setMinimum(min)
1030 {
        if ( min > this.maximum )
        {
           throw new CJException.InvalidValue(
              "minimum value " + min + " must be <= the maximum value of "+this.maximum+"." );
1035    }
        this.minimum=min;
     }


1040 /************************************************************************
     Class : PrimitiveNumber
     Method: getResNumerator, hasResNumerator and setResNumerator
     Params: varies
     Return: varies
1045 Description: One restriction for the PrimitiveNumber is the resolution
                  of it's numerator these methods handle this restriction.
     ************************************************************************/
     function _PrimitiveNumber_getResNumerator()
     {
1050    if(this.resnumerator==null) return 1.0;
        else return this.resnumerator;
     }

     function _PrimitiveNumber_hasResNumerator()
1055 {
        return this.resnumerator != null;
```

```
    }

    function _PrimitiveNumber_setResNumerator(resnum )
1060 {
       if ( resnum < 0 )
       {
          throw new CJException.InvalidValue("ResNumerator " + resnum + " must be >= 0" );
       }
1065   this.resnumerator=resnum;
    }

    /**************************************************************************
     Class : PrimitiveNumber
1070 Method: getResDenominator, hasResDenominator and setResDenominator
     Params: varies
     Return: varies
     Description: One restriction for the PrimitiveNumber is the resolution
                  of it's denominator these methods handle this restriction.
1075 **************************************************************************/
    function _PrimitiveNumber_getResDenominator()
    {
       if(this.resdenominator==null)
          return 1.0;
1080   else
          return this.resdenominator;
    }

    function _PrimitiveNumber_hasResDenominator()
1085 {
       return this.resdenominator != null;
    }

    function _PrimitiveNumber_setResDenominator(resdenom )
1090 {
       if ( resdenom < 0 )
       {
          throw new CJException.InvalidValue("ResDenominator " + resdenom + " must be >= 0" );
       }
1095
       this.resdenominator=resdenom;
    }

    /**************************************************************************
1100 Class : PrimitiveNumber
     Method: getTolerance, hasTolerance and setTolerance
     Params: varies
```

```
     Return: varies
     Description: One restriction for the PrimitiveNumber is the resolution
1105            of it's numerator these methods handle this restriction.
     ************************************************************************/
     function _PrimitiveNumber_getTolerance()
     {
        if(this.tolerance==null)
1110       return 0.0;
        else
           return this.tolerance;
     }

1115 function _PrimitiveNumber_hasTolerance()
     {
        return this.tolerance != null;
     }

1120 function _PrimitiveNumber_setTolerance(tol )
     {
        if ( tol < 0 )
        {
           throw new CJException.InvalidValue("Tolerance " + tol + " must be >= 0" );
1125    }
        this.tolerance=tol;
     }

     /************************************************************************
1130  Class : PrimitiveNumber
     Method: getPrecision, hasPrecision and setPrecision
     Params: varies
     Return: varies
     Description: One restriction for the PrimitiveNumber is it's precision
1135            these methods handle this restriction.
     ************************************************************************/
     function _PrimitiveNumber_getPrecision()
     {
        if(this.precision==null)
1140       return Number.MAX_VALUE
        else
           return this.precision;
     }

1145 function _PrimitiveNumber_hasPrecision()
     {
        return this.precision != null;
     }
```

```
function _PrimitiveNumber_setPrecision(prec )
{
   if ( prec < 0 )
   {
      throw new CJException.InvalidValue("Precision " + prec + " must be >= 0" );
   }
   this.precision=prec;
}


/*********************************************************************
 Class : PrimitiveNumber
 Method: validateValue
 Params: Number test
 Return: null
 Description: checks the test value against the primitive's restrictions
*********************************************************************/
function _PrimitiveNumber_validateValue( test )
{
   errorString="";

   // Convert to a number.
   test = new Number(test);

   if (isNaN(test) == true)
   {
      errorString += "\"" + test + "\" is not a number.";
      throw new CJException.InvalidValue(errorString);
   }

   //check if the number is between the min and max allowable values
   if ( this.hasMaximum() )
   {
      if( test > this.getMaximum() )
         errorString+="The Number "+ test +" exceeds limit of "+ this.getMaximum() +".\n";
   }
   if ( this.hasMinimum() )
   {
      if( test < this.getMinimum() )
         errorString+="The Number "+ test +" is below limit of "+ this.getMinimum() +".\n";
   }
  //if only the resnumerator is set treat as an integer resolution
   if ( this.hasResNumerator() && !this.hasResDenominator() )
   {
      res = this.getResNumerator();
```

```
1195        min = ( this.hasMinimum() ? this.getMinimum() : 0 );
            baseVal = test - min;

            if ( baseVal % res != 0 )
            {
1200           errorString+="value \"" + test + "\" has an invalid resolution (" +
                        res + ")\n";
            }
         }

1205     // if both the resnumerator and the res denominator are set
         //treat as a fractional resolution
         if (`this.hasResNumerator() && this.hasResDenominator() )
         {
            min = ( this.hasMinimum() ? this.getMinimum() : 0.0 );
1210        num = this.getResNumerator();
            den = this.getResDenominator();
            testVal = test - min;

            // Determine nearest integral number of units of precision
1215        // (ex: if res=1/3, then 2.666 is 8 "thirds", so testVal=8
            testVal = (testVal * den) / num;

            // Now determine what the proper value should be
            // (ex: for testVal=8, we end up with 8/3, or 2.6666..
1220        // and determine the delta from that to compare against tolerance.
            // Note: You can't just take the value above since "0.34"
            // would result in 1.02, which would appear to be invalid
            // for a tolerance of 0.01.  So recompute what the exact value
            // should be, and compare our tolerance to that.
1225        testVal = min + ( Math.round(testVal) * num / den );

            if ( Math.abs( testVal - test) > this.getTolerance() )
            {
               errorString+="value \"" + test + "\" has an invalid resolution (" +
1230              num + "/" + den + "+-" + this.getTolerance() + ")\n";
            }
         }

         // now check number of significant digits
1235     if ( this.hasPrecision() )
         {
            exp = Math.pow( 10.0, this.getPrecision() );
            testVal = test * exp;
            if ( (test - Math.floor(testVal)/exp) > 0.0 )
1240        {
```

```
            errorString+="value \"" + test + "\" may only have " + this.getPrecision()
                  + " significant digits.\n";
         }
      }
1245
      if(errorString!="")
      {
         throw new CJException.InvalidValue(errorString);
      }
1250 }

   PrimitiveNumber.prototype = new PrimitiveBase();
   PrimitiveNumber.prototype.fromString        = _PrimitiveNumber_fromString;
   PrimitiveNumber.prototype.validateValue     = _PrimitiveNumber_validateValue;
1255 PrimitiveNumber.prototype.maximum         = null;
   PrimitiveNumber.prototype.getMaximum        = _PrimitiveNumber_getMaximum;
   PrimitiveNumber.prototype.hasMaximum        = _PrimitiveNumber_hasMaximum;
   PrimitiveNumber.prototype.setMaximum        = _PrimitiveNumber_setMaximum;
   PrimitiveNumber.prototype.minimum           = null;
1260 PrimitiveNumber.prototype.getMinimum        = _PrimitiveNumber_getMinimum;
   PrimitiveNumber.prototype.hasMinimum        = _PrimitiveNumber_hasMinimum;
   PrimitiveNumber. CLASS: PrimitiveSetElement

      An enumeration/bitstring has an associated "PrimitiveSetDefinition"
1265  that contains a list of "PrimitivedSetElement" objects.  A set
      element is simply a key (typically a number) associated with a value.
      The key ('num') must be a string or a number.
      The value may be any JavaScript object and care must be taken to
      ensure the user of the value is aware of the value's type.
1270
   ************************************************************************
   ************************************************************************/

   // and number and associated value
1275 function PrimitiveSetElement( num, val )
   {
      this.num = num;
      this.val = val;
   }
1280




1285
   /************************************************************************
```

```
    ************************************************************************

     CLASS: PrimitiveSetDefinition
1290
      A PrimitiveSetDefinition is a container of an array of
      PrimitiveSetElements.  The elements are an ordered list.
     Restrictions:
        none
1295
    ************************************************************************
    ************************************************************************/

    function PrimitiveSetDefinition( id, writable, elements )
1300 {
       this.PrimitiveBase( id, writable,  elements );
1325 }

    /************************************************************************
     Class : PrimitiveSetDefinition
     Method: getElementText
1330  Params: val
     Return: String
     Description: Returns the string representation of the given element
       value.  This value must match once of the enumeration values.
     Exceptions: NotFound
1335 ************************************************************************/
    function _PrimitiveSetDefinition_getElementText( val )
    {
       if ( this.data != null )
       {
1340      for ( var i=0; i<this.data.length; i++ )
          {
             if ( val == this.data[i].num )
                return this.data[i].val.toString();
          }
1345    }
       throw new CJException.NotFound(
             "PrimitiveSetDefinition_getElementText - element for value \""
             + val + "\"  in set \"" + this.id + "\" not found" );
    }
1350
    /************************************************************************
     Class : PrimitiveSetDefinition
     Method: getElementIndex
     Params: val
1355  Return: Number
```

```
     Description: Returns the index into the array of enumeration elements
       of the specified value.
    ************************************************************************/
    function _PrimitiveSetDefinition_getElementIndex( val )
1360 {
       if ( this.data != null )
       {
          for ( var i=0; i<this.data.length; i++ )
          {
1365         if ( val == this.data[i].num )
                return i;
          }
       }
       throw new CJException.NotFound(
1370          "PrimitiveSetDefinition_getElementIndex - element for value \""
              + val + "\"  in set \"" + this.id + "\" not found" );
    }


1375 PrimitiveSetDefinition.prototype = new PrimitiveBase();
    PrimitiveSetDefinition.prototype.getElementText  = _PrimitiveSetDefinition_getElementText;
    PrimitiveSetDefinition.prototype.getElementIndex = _PrimitiveSetDefinition_getElementIndex;

    /************************************************************************
1380 ************************************************************************

      CLASS: PrimitiveEnum

        A PrimitiveEnum is a enumeration value and an assocaited
1385    PrimitiveSetDefinition.  The value of the PrimitiveEnum must always
        be one of the assoication set definition values.

        The class's data type is a Number.

1390    Restrictions:
          allowed - string of numbers separated by commas, indicating which of the
                    values in the definition are allowed.  Whitespace is not
                    permitted in the 'allowed' string.  This restriction may be
                    'null' indicating that there is no restriction on which
1395                definition element may be selected.  (null=any)

    ************************************************************************
    ************************************************************************/

1400 function PrimitiveEnum( id, writable,  initIndex, setId, allowed )
    {
```

```
      this.setAllowed(allowed);
      // findPrimitive will throw exception if ID is bad
      this.enumDef = PrimitiveBase.findPrimitive( setId );

      this.enumDef.addListener( this );

      this.PrimitiveBase( id, writable,  initIndex );
   }

   /**************************************************************************
    Class : PrimitiveEnum
    Method: valueChanged
    Params: prim
    Return: none
    Description: A PrimitiveEnum is a listener of it PrimitiveSetDefinition.
      Should the definition set change, then the representation of the
      value should also change.  This does nothing more than propagate a
      change to the enum's listeners.
   **************************************************************************/
   function _PrimitiveEnum_valueChanged( prim )
   {
      // propagate change
      this.notifyListeners( this );
   }



   // return set definition
   function _PrimitiveEnum_getSetDefinition( )
   {
      return this.enumDef;
   }

   // returned element's enum text
   function _PrimitiveEnum_getElementText( )
   {
      return this.enumDef.getElementText( this.data );
   }

   // return element's enum index
   function _PrimitiveEnum_getElementIndex( )
   {
      return this.enumDef.getElementIndex( this.data );
   }

   /**************************************************************************
```

```
     Class : PrimitiveEnum
     Method: increment
1450 Params: Listener from
     Return: Boolean - if changed
     Description: Increments the enumeration value to the next value in
       the set defition.  Returns false if already at the last element.
    ************************************************************************/
1455 function _PrimitiveEnum_increment( from )
    {
       if ( this.enumDef.data == null )
          return false;

1460   var n = this.getElementIndex();
       if ( n+1 < this.enumDef.data.length )
       {
          this.setValue( this.enumDef.data[n+1].num, from );
          return true;
1465   }
       return false;
    }

    /************************************************************************
1470 Class : PrimitiveEnum
     Method: decrement
     Params: Listener from
     Return: Boolean - if changed
     Description: Decrements the enumeration value to the previous value in
1475   the set defition.  Returns false if already at the first element.
    ************************************************************************/
    function _PrimitiveEnum_decrement( from )
    {
       if ( this.enumDef.data == null )
1480      return false;

       var n = this.getElementIndex();
       if ( n > 0 )
       {
1485      this.setValue( this.enumDef.data[n-1].num, from );
          return true;
       }
       return false;
    }
1490
    /************************************************************************
     Class : PrimitiveEnum
     Method: fromString
```

```
     Params: String text
     Return: Number
     Description: converts the string given into a number
    ************************************************************************/
    function _PrimitiveEnum_fromString(text)
    {
       if(text==null){ return null;}
       //parse out the first integer in the string.
       //must begin with an number.
       var number=parseInt(text);
       //if a number was found return it
       if(!isNaN(number)){ return number;}

       // else return the unchanged text
       return text;
    }

    /************************************************************************
     Class : PrimitiveEnum
     Method: getAllowed, hasAllowed and setAllowed
     Params: varies
     Return: varies
     Description: a restriction for the PrimitiveEnum is which enums are
               allowed these methods handle this restriction.
    ************************************************************************/
    function _PrimitiveEnum_getAllowed()
    {
       return this.allowed;
    }

    function _PrimitiveEnum_hasAllowed()
    {
       return this.allowed != null;
    }

    function _PrimitiveEnum_setAllowed(allowed )
    {
       // Allow ANY 'allowed' string since it may now be text too.
       this.allowed=allowed;
    }

    /************************************************************************
     Class : PrimitiveEnum
     Method: valuedateValue
     Params: Object test
     Return: none
```

```
1540  Description: Overrides the default validateValue() method.
        If there are no restrictions, then an implicit restriction is that
        it must be one of the number in the set definition
      ************************************************************************/
      function _PrimitiveEnum_validateValue( test )
1545  {
        var allowed = this.getAllowed();
        if ( allowed != null )
        {
          // search for allowed value by looking for ",3," in ",1,2,3,4,5,"
1550      // note that this will break if spaces are present
          allowed = "," + allowed + ",";
          var searchVal = "," + test + ",";
          if ( allowed.indexOf( searchVal ) == -1 )
          {
1555        throw new CJException.InvalidValue("Enumerated value \"" + test +
                    "\" is disallowed, subset is \"" + this.getAllowed() + "\"" );
          }
        }
        //check if the test value exists in enumeration
1560    if ( this.data != null )
        {
          for ( var i=0; i<this.enumDef.data.length; i++ )
          {
            if ( this.enumDef.data[i].num == test )
1565        {
              this.base_validateValue( test );
              return;
            }
          }
1570    }
        throw new CJException.InvalidValue(
              "Value \"" + test + "\" for PrimitiveEnum \"" + this.id
              + "\" is not in definition" );
      }
1575

      PrimitiveEnum.prototype = new PrimitiveBase();
      PrimitiveEnum.prototype.valueChanged      = _PrimitiveEnum_valueChanged;
      PrimitiveEnum.prototype.base_validateValue= _PrimitiveBase_validateValue;
1580  PrimitiveEnum.prototype.getSetDefinition  = _PrimitiveEnum_getSetDefinition;
      PrimitiveEnum.prototype.getElementText    = _PrimitiveEnum_getElementText;
      PrimitiveEnum.prototype.getElementIndex   = _PrimitiveEnum_getElementIndex;
      PrimitiveEnum.prototype.increment         = _PrimitiveEnum_increment;
      PrimitiveEnum.prototype.decrement         = _PrimitiveEnum_decrement;
1585  PrimitiveEnum.prototype.fromString        = _PrimitiveEnum_fromString;
```

```
    PrimitiveEnum.prototype.validateValue     = _PrimitiveEnum_validateValue;
    PrimitiveEnum.prototype.allowed           = null;
    PrimitiveEnum.prototype.getAllowed        = _PrimitiveEnum_getAllowed;
    PrimitiveEnum.prototype.hasAllowed        = _PrimitiveEnum_hasAllowed;
    PrimitiveEnum.prototype.setAllowed        = _PrimitiveEnum_setAllowed;



/*************************************************************************
 *************************************************************************

  CLASS: PrimitiveOctetString

    A PrimitiveOctetString is a sequence of octets (bytes) stored as
    a hexidecimal string.  It's data type is String.  There are helper
    methods to get and set individual bytes, but it is recommended that
    the content typical be set as a whole.

    The class's data type is a String.

   Restrictions:
      maxlength - max octet length.  This the stored string length may be not
                  more that twice this value since there are two characters
                  required per byte. (null = no limit)

*************************************************************************
*************************************************************************/
function PrimitiveOctetString( id, writable, initData, maxlength)
{
   this.setMaxLength(maxlength);
   this.PrimitiveBase( id, writable, initData );
}

/*************************************************************************
 Class : PrimitiveOctetString
 Method: getLength
 Params:
 Return: number of octets in the string
 Description: This method returns the number of octets in the string.
   Which is exactly half of the string length.
*************************************************************************/
function _PrimitiveOctetString_getLength( )
{
   return ( ( this.data == null ) ? 0 : this.data.length / 2 );
}
```

```
    /**************************************************************************
     Class : PrimitiveOctetString
1635  Method: getByte
     Params: Number position
     Return: Number (byte)
     Description: Returns the octet (byte) at the specified byte offset.
     Exceptions: OutOfRange
1640 **************************************************************************/
    function _PrimitiveOctetString_getByte( pos )
    {
       if ( pos < this.getLength() )
       {
1645      return parseInt( this.data.substr( pos * 2, 2 ), 16 );
       }
       throw new CJException.OutOfRange(
          "PrimitiveOctetString_getByte - byte position \"" + pos
          + "\" invalid, range is 0 to " + this.getLength() + "-1" );
1650 }

    /**************************************************************************
     Class : PrimitiveOctetString
     Method: setByte
1655  Params: Number position, Number octet(byte)
     Return: none
     Description: Sets the specified octet (byte) at the specified offset.
     Exceptions: OutOfRange
    **************************************************************************/
1660 function _PrimitiveOctetString_setByte( pos, theByte, from )
    {
       if ( pos < this.getLength() )
       {
          theByte = theByte & 0xff;
1665      var insert = theByte.toString( 16 );
          if ( insert.length == 1 )
          {
             insert = "0" + insert;
          }
1670
          // this is a pain to replace a segment of a string...
          var tmp = this.data.substring( 0, pos*2 );
          tmp += insert;
          tmp += this.data.substring( pos*2+2, this.getLength()*2 );
1675
          return this.setValue( tmp, from );
       }
```

```
      throw new CJException.OutOfRange(
         "PrimitiveOctetString_setByte - byte position \"" + pos
1680     + "\" invalid, range is 0 to " + this.getLength() + "-1" );
   }

   /**************************************************************************
    Class : PrimitiveOctetString
1685 Method: getMaxLength, hasMaxLength and setMaxLength
    Params: varies
    Return: varies
    Description: The restriction for the PrimitiveOctetString is its length
           these methods handle that restriction.
1690 **************************************************************************/
   function _PrimitiveOctetString_getMaxLength()
   {
      if ( this.maxlength == null )
         return Number.MAX_VALUE;
1695  else
         return this.maxlength;
   }

   function _PrimitiveOctetString_hasMaxLength()
1700 {
      return this.maxlength != null;
   }

   function _PrimitiveOctetString_setMaxLength(maxlen)
1705 {
      if ( maxlen < 0 )
      {
         throw new CJException.InvalidValue(
            "max length " + maxlen + " must be >= 0" );
1710  }
      this.maxlength=maxlen;
   }


1715 /**************************************************************************
    Class : PrimitiveOctetString
    Method: validateValue
    Params: OctetString test
    Return: null
1720 Description: checks the test value against the primitive's restrictions
   **************************************************************************/
   function _PrimitiveOctetString_validateValue( test )
   {
```

```
        var errorString="";
1725    //check the test value's length.
        if ( this.hasMaxLength() && test.length > (this.getMaxLength() * 2) )
        {
              errorString+="Octet string with " + test.length/2 + " octets exceeds limit of "
                          + this.getMaxLength() + " octets (2 digit hex pairs)\n";
1730    }
        //check if the octet is in pairs
        if ( test.length % 2 != 0 )
        {
           errorString+="Octet string of length " + test.length + " must be multiple of 2\n";
1735    }
        //check for invalid characters (ie not 0-F) case insensitive on letters
        var badChar=test.search(/[^0-9A-F]/i);
        if(badChar!=-1)
        {
1740       errorString+="Octet string \"" + test +
                    "\" has invalid character \""+test.charAt(badChar)+"\" at position " + badChar;
        }
        if(errorString!="")
        {
1745       throw new CJException.InvalidValue(errorString);
        }
    }

    PrimitiveOctetString.prototype = new PrimitiveBase();
1750 PrimitiveOctetString.prototype.getLength    = _PrimitiveOctetString_getLength;
    PrimitiveOctetString.prototype.getByte      = _PrimitiveOctetString_getByte;
    PrimitiveOctetString.prototype.setByte      = _PrimitiveOctetString_setByte;
    PrimitiveOctetString.prototype.setByte      = _PrimitiveOctetString_setByte;

    PrimitiveOctetString.prototype.validateValue= _PrimitiveOctetString_validateValue;
1755 PrimitiveOctetString.prototype.maxlength    = null;
    PrimitiveOctetString.prototype.getMaxLength = _PrimitiveOctetString_getMaxLength;
    PrimitiveOctetString.prototype.hasMaxLength = _PrimitiveOctetString_hasMaxLength;
    PrimitiveOctetString.prototype.setMaxLength = _PrimitiveOctetString_setMaxLength;

1760

    /*****************************************************************************
     *****************************************************************************

1765    CLASS: PrimitiveBacnetOID

        The PrimitiveBacnetOID is a representation of a Bacnet object ID.
        It's data representation is a Number that is 32 bits.  The high
```

```
   order 10 bits is the bacnet object type, and the low order 22 bits
   is the instance number.

   The class's data type is a Number.

*************************************************************************
*************************************************************************/
function PrimitiveBacnetOID( id, writable, initData )
{
   this.PrimitiveBase( id, writable, initData );
}

// get object type (high 10 bits)
function _PrimitiveBacnetOID_getType( )
{
   return ( parseInt(this.data) );
}

// get object instance (low 22 bits)
function _PrimitiveBacnetOID_getInstance( )
{
   return ( parseInt(this.data.substr(this.data.indexOf(":")+1) ) );
}

// set object type (high 10 bits)
function _PrimitiveBacnetOID_setType( type, from )
{
   return this.setValue( type + ":" + this.getInstance(), from );
}

// set object instance (low 22 bits)
function _PrimitiveBacnetOID_setInstance( instance, from )
{
   return this.setValue( this.getType() + ":" + instance, from );
}

/*************************************************************************
 Class : PrimitiveBacnetOID
 Method: fromString
 Params: String text
 Return: Number
 Description: converts the string given into a bacnetOID in integer form
*************************************************************************/
function _PrimitiveBacnetOID_fromString(text)
{
   if (text==null)
```

```
    {
       return "-1:-1";
    }

    //first integer is the type
    var type=parseInt(text);

    //integer after ':' is the instance
    var instance=parseInt(text.substr(text.indexOf(":")+1));

    //if both numbers were successfully parsed then return the bacnetOID
    if(!isNaN(type) && !isNaN(instance))
    {
       return text;
    }

    //throw an exception if the conditions above were not met
    throw new CJException.InvalidValue("\nPrimitiveBacnetOID: fromString() must be given a string"+
                                       "\nof the form \"decimal:decimal\". It received \""+text+"\".");
}

/**********************************************************************
 Class : PrimitiveBacnetOID
 Method: toString
 Params:
 Return: String
 Description: bacnetOID in String form "type:instance"
**********************************************************************/
function _PrimitiveBacnetOID_toString()
{
       //There is a little bit of a debate over what this should return
    //when the value is null. For now we are returning the empty string "".
    if(this.data==null)
    {
       return "-1:-1";
    }
    else
    {
       return this.data;
    }
}

PrimitiveBacnetOID.prototype = new PrimitiveBase();
PrimitiveBacnetOID.prototype.getType          = _PrimitiveBacnetOID_getType;
PrimitiveBacnetOID.prototype.getInstance      = _PrimitiveBacnetOID_getInstance;
```

```
PrimitiveBacnetOID.prototype.setType          = _PrimitiveBacnetOID_setType;
PrimitiveBacnetOID.prototype.setInstance      = _PrimitiveBacnetOID_setInstance;
PrimitiveBacnetOID.prototype.toString         = _PrimitiveBacnetOID_toString;
PrimitiveBacnetOID.prototype.fromString       = _PrimitiveBacnetOID_fromString;
1865



1870
/*************************************************************************
**************************************************************************

  CLASS: PrimitiveDate
1875
  A PrimitiveDate is a 4 octet value containing a year, month, day,
  and day of week.  The year is based from 1900.  Any field may be
  0xFF indicating an unspecified value.

1880  The class's data type is a Number.

**************************************************************************
**************************************************************************/
function PrimitiveDate( id, writable, initData )
1885 {
   this.PrimitiveBase( id, writable, initData );
}

// Note - if this.data is 'null', you get 0 for everything
1890 function _PrimitiveDate_getYear()
   { return (( this.data & 0xff000000) >>> 24 ); }

function _PrimitiveDate_getMonth()
   { return (( this.data & 0x00ff0000) >>> 16 ); }
1895
function _PrimitiveDate_getDay()
   { return (( this.data & 0x0000ff00) >>>  8 ); }

function _PrimitiveDate_getDayOfWeek()
1900   { return (( this.data & 0x000000ff) ); }

function _PrimitiveDate_setYear( v, from )
   { this.setValue((this.data & 0x00ffffff) + (v<<24), from);}

1905 function _PrimitiveDate_setMonth( v, from )
   { this.setValue((this.data & 0xff00ffff) + (v<<16), from);}
```

```
   function _PrimitiveDate_setDay( v, from )
      { this.setValue((this.data & 0xffff00ff) + (v<< 8), from);}
1910
   function _PrimitiveDate_setDayOfWeek( v, from )
      { this.setValue((this.data & 0xffffff00) + (v), from ); }

   // return the primitive's value
1915 function _PrimitiveDate_getValue()
   {
      //if the value appears negative change it to the positive value
      if(this.data<0) return 0x100000000+this.data
      return this.data;
1920 }

   /*********************************************************************
    Class : PrimitiveDate
    Method: fromString
1925  Params: String text
    Return: Number
    Description: converts the date string given into an integer form.
   Uses the internet standard "GMT"  format date as produced by the java
   Date.toGMTString(). The format is "Sat 12 Aug 1995".  Note capitalization.
1930 *********************************************************************/
   function _PrimitiveDate_fromString(text)
   {
      if(text==null){ return null;}
      var dayOfWeek;
1935  var day;
      var month;
      var year;

      //split the text string at every place there is one or more spaces.
1940  //this creates an array with all the separated elements
      var date=text.split(/ +/);

      //test for improperly formated date
      CJAssert(date.length==4,"\nPrimitiveDate: fromString() must be given a string "+
1945                           "\nof the form \"Sat 12 Aug 1995\". It received \""+text+"\","+
                               "\nwhich has too many spaces.");

      //first element is the day of the week
      for(var loop=0; loop<PrimitiveDate.dayArray.length;loop++)
1950  {
         if(date[0].toLowerCase()==PrimitiveDate.dayArray[loop].toLowerCase()) break;
      }
```

```
    if(loop>=PrimitiveDate.dayArray.length){dayOfWeek==null;}
    else
1955 {
       if(loop==0){dayOfWeek=0xFF;}
       else{dayOfWeek=loop;}
    }

1960 //second element is the day of the month,
    //replaces every occurance of "*" with "255" (ie 0xFF) before parsing
    //The internal form of the date uses 0xFF as the wild card value
    day=parseInt(date[1].replace(/\*/g, "255"));
    if(day<0||day>255){day=NaN;}
1965
    //third element is the month
    for(loop=0; loop<PrimitiveDate.monthArray.length;loop++)
    {
       if(date[2].toLowerCase()==PrimitiveDate.monthArray[loop].toLowerCase()) break;
1970 }
    if(loop>=PrimitiveDate.monthArray.length){month==null;}
    else
    {
       if(loop==0){month=0xFF;}
1975   else{month=loop;}
    }

    //fourth element is the year
    //replaces every occurance of "*" with "2155" (ie 255+1900) before parsing
1980 //subtract 1900 to get the year to be stored, must be 0-255
    year=parseInt(date[3].replace(/\*/g, "2155"))-1900;
    if(year<0||year>255){year=NaN;}

    //throw an exception if any of the elements did not parse correctly
1985 CJAssert(dayOfWeek!=null || !isNaN(day) || month!=null || !isNaN(year),
                "\nPrimitiveDate: fromString() must be given a string "+
                "\nof the form \"Sat 12 Aug 1995\". It received \""+text+"\".");

    var data;
1990 data=(data & 0x00ffffff) + (year<<24);
    data=(data & 0xff00ffff) + (month<<16);
    data=(data & 0xffff00ff) + (day<< 8);
    data=(data & 0xffffff00) + (dayOfWeek);

1995 return data;

  }
  /*************************************************************************
```

```
     Class : PrimitiveDate
2000 Method: toString
     Params:
     Return: String
     Description: returns the string form of the Date.
                  The form is "www D mmm YYYY" ex. "Sat 12 Aug 1995"
2005 ************************************************************************/
     function _PrimitiveDate_toString()
     {
        //There is a little bit of a debate over what this should return
        //when the value is null. For now we are returning the empty string "".
2010    if(this.data==null){return "";}

       //looks up the proper dayOfWeek and month in the respective arrays
       //and gets the day and year. Then concatenates them together in the
       //correct order separated by spaces.
2015   var dayOfWeek=this.getDayOfWeek();
       var day=this.getDay();
       var month=this.getMonth();
       var year=this.getYear();

2020   if(dayOfWeek==0xFF){dayOfWeek=0;}
       dayOfWeek=PrimitiveDate.dayArray[dayOfWeek];

       if(month==0xFF){month=0;}
       month=PrimitiveDate.monthArray[month];
2025
       if(day==0xFF){day="*";}

       if(year==0xFF){year="*";}
       else{year=year+1900}
2030
       return (dayOfWeek+" "+day+" "+month+" "+year);
     }

     PrimitiveDate.dayArray=new Array("*", "Mon", "Tue", "Wed",
2035                                  "Thu", "Fri", "Sat", "Sun");
     PrimitiveDate.monthArray=new Array("*", "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");

     PrimitiveDate.prototype = new PrimitiveBase();
2040 PrimitiveDate.prototype.getYear      = _PrimitiveDate_getYear;
     PrimitiveDate.prototype.getMonth     = _PrimitiveDate_getMonth;
     PrimitiveDate.prototype.getDay       = _PrimitiveDate_getDay;
     PrimitiveDate.prototype.getDayOfWeek = _PrimitiveDate_getDayOfWeek;
     PrimitiveDate.prototype.setYear      = _PrimitiveDate_setYear;
```

```
PrimitiveDate.prototype.setMonth     = _PrimitiveDate_setMonth;
PrimitiveDate.prototype.setDay       = _PrimitiveDate_setDay;
PrimitiveDate.prototype.setDayOfWeek = _PrimitiveDate_setDayOfWeek;
PrimitiveDate.prototype.toString     = _PrimitiveDate_toString;
PrimitiveDate.prototype.fromString   = _PrimitiveDate_fromString;
PrimitiveDate.prototype.getValue     = _PrimitiveDate_getValue;
/*************************************************************************
**************************************************************************

  CLASS: PrimitiveTime

  A PrimitiveTime is a 4 octet value containing a hour, minute, second
  and 100ths of a second.  Any field may be 0xFF indicating an
  unspecified value.

  The class's data type is a Number.

**************************************************************************
**************************************************************************/
function PrimitiveTime( id, writable, initData )
{
   this.PrimitiveBase( id, writable, initData );
}

// Note - if this.data is 'null', you get 0 for everything
function _PrimitiveTime_getHour()
   { return (( this.data & 0xff000000) >>> 24 ); }

function _PrimitiveTime_getMinute()
   { return (( this.data & 0x00ff0000) >>> 16 ); }

function _PrimitiveTime_getSecond()
   { return (( this.data & 0x0000ff00) >>>  8 ); }

function _PrimitiveTime_get100ths()
   { return (( this.data & 0x000000ff) ); }

function _PrimitiveTime_setHour( v, from )
   { this.setValue((this.data & 0x00ffffff) + (v<<24), from );}

function _PrimitiveTime_setMinute( v, from )
   { this.setValue((this.data & 0xff00ffff) + (v<<16), from );}

function _PrimitiveTime_setSecond( v, from )
   { this.setValue((this.data & 0xffff00ff) + (v<< 8), from );}
```

```
   function _PrimitiveTime_set100ths( v, from )
      { this.setValue((this.data & 0xffffff00) + (v), from  ); }

    // return the primitive's value
2095 function _PrimitiveTime_getValue()
   {
      //if the value appears negative change it to the positive value
      if(this.data<0) return 0x100000000+this.data
      return this.data;
2100 }
   /**************************************************************************
    Class : PrimitiveTime
    Method: fromString
    Params: String text
2105  Return: Number
    Description: converts the Time string given into an integer form.
            The string is of the form "h:mm:ss.HH",
            where h is hours (one or 2 digits), mm is minutes (2 digits),
            ss is seconds (2 digits), and HH is hundredths (2 digits).
2110        Any one of these fields can be the single char '*' for wild card.

   **************************************************************************/
   function _PrimitiveTime_fromString(text)
   {
2115  if(text==null){ return null;}
      var hour;
      var minute;
      var second;
      var hundredth;
2120
      //replaces every occurance of "*" with "255" (ie 0xFF)
      //The internal form of the time uses 0xFF as the wild card value
      var time=text.replace(/\*/g, "255")

2125  //split the text string at every place there is a ":".
      //this creates an array with all the separated elements
      time=time.split(/:/);

      //test for improperly formated time
2130  CJAssert(time.length==4,"\nPrimitiveTime: fromString() must be given a string "+
                              "\nof the form \"HH:MM:SS:hh\". It received \""+text+"\","+
                              "\nwhich has too many \":\"'s");

      //first element is the hour
2135  hour=parseInt(time[0]);
      if(hour<0||hour>255){hour=NaN;}
```

```
        //second element is the minutes
        minute=parseInt(time[1]);
2140    if(minute<0||minute>255){minute=NaN;}

        //third element is the seconds
        second=parseInt(time[2]);
        if(second<0||second>255){second=NaN;}
2145
        //fourth element is the get100ths of a second
        hundredth=parseInt(time[3]);
        if(hundredth<0||hundredth>255){hundredth=NaN;}

2150    //throw an exception if any of the elements did not parse correctly
        CJAssert(!isNaN(hour) || !isNaN(minute) || !isNaN(second) || !isNaN(hundredth),
                            "\nPrimitiveTime: fromString() must be given a string "+
                            "\nof the form \"HH:MM:SS:hh\". It received \""+text+"\".");

2155    var data;
        data=(data & 0x00ffffff) + (hour<<24);
        data=(data & 0xff00ffff) + (minute<<16);
        data=(data & 0xffff00ff) + (second<< 8);
        data=(data & 0xffffff00) + (hundredth);
2160
        return data;

    }
    /**************************************************************************
2165 Class : PrimitiveTime
     Method: toString
     Params:
     Return: String
     Description: returns the string form of the Date.
2170              The form is "HH:MM:SS:hh" ex. "23:59:59:99"
                  uses the 24 hour clock and the minutes, seconds, and 100s
                  have leading zeros if <10;
    **************************************************************************/
    function _PrimitiveTime_toString()
2175{

       //There is a little bit of a debate over what this should return
       //when the value is null. For now we are returning the empty string "".
       if(this.data==null){return "";}
2180
      //gets the time values. If they are 0xFF make them the wildcard "*" otherwise
      //if they are only one digit add a leading 0. Hour does not have that.
```

```
    //Then concatenate them together in the correct order separated by spaces.
    var hour=this.getHour();
2185 var minute=this.getMinute();
    var second=this.getSecond();
    var hundredths=this.get100ths();

    if(hour==0xFF){dayOfWeek="*";}
2190
    if(minute==0xFF){minute="*";}
    else{minute=minute<10?"0"+minute:minute}

    if(second==0xFF){second="*";}
2195 else{second=second<10?"0"+second:second}

    if(hundredths==0xFF){hundredths="*";}
    else{hundredths=hundredths<10?"0"+hundredths:hundredths}

2200 return (hour+":"+minute+":"+second+":"+hundredths);
  }


  PrimitiveTime.prototype = new PrimitiveBase();
2205 PrimitiveTime.prototype.getHour      = _PrimitiveTime_getHour;
  PrimitiveTime.prototype.getMinute    = _PrimitiveTime_getMinute;
  PrimitiveTime.prototype.getSecond    = _PrimitiveTime_getSecond;
  PrimitiveTime.prototype.get100ths    = _PrimitiveTime_get100ths;
  PrimitiveTime.prototype.setHour      = _PrimitiveTime_setHour;
2210 PrimitiveTime.prototype.setMinute    = _PrimitiveTime_setMinute;
  PrimitiveTime.prototype.setSecond    = _PrimitiveTime_setSecond;
  PrimitiveTime.prototype.set100ths    = _PrimitiveTime_set100ths;
  PrimitiveTime.prototype.toString     = _PrimitiveTime_toString;
  PrimitiveTime.prototype.fromString   = _PrimitiveTime_fromString;
2215 PrimitiveTime.prototype.getValue     = _PrimitiveTime_getValue;


  /**************************************************************************
2220  **************************************************************************

    CLASS: PrimitiveBitString

    The PrimitiveBitString is an array of bit values stored as a string,
2225  of ones and zeros, e.g. "011010010010". Most significant bit is first.

    The class's data type is a String.
```

```
      Restrictions:
2230     maxbits - max # of bits allowed in bit string (no value = infinite)
         allowed - string of '1' and '0' characters where '1' indicates that the
                   bit position is allowed to be set.  This restriction may be
                   'null' indicating that there is no restriction on which bits
                   may be set.  (null=any)
2235 ************************************************************************
     ************************************************************************/
     function PrimitiveBitString( id, writable, initData, defID, maxbits, allowed )
     {
        this.setMaxBits(maxbits);
2240    this.setAllowed(allowed);
        if ( defID != null )
        {
           this.enumDef = PrimitiveBase.findPrimitive( defID );
           this.enumDef.addListener( this );
2245     }

        this.PrimitiveBase( id, writable, initData );
     }

2250 /************************************************************************
      Class : PrimitiveBitString
      Method: valueChanged
      Params: prim
      Return: none
2255  Description: A PrimitiveBitString is a listener of it's
        PrimitiveSetDefinition.  Should the definition set change, then
        the representation of the bit values should also change.  This does
        nothing more than propagate a change to the prim's listeners.
     ************************************************************************/
2260 function _PrimitiveBitString_valueChanged( changedDef )
     {
        // propagate change
        this.notifyListeners( this );
     }
2265

     /************************************************************************
      Class : PrimitiveBitString
      Method: getLength
2270  Params:
      Return: Number of bits
      Description: Returns the number of defined bits in the bit string.
     ************************************************************************/
     function _PrimitiveBitString_getLength( )
```

```
{
   if ( this.data == null ){return 0;}
   else{return this.data.length;}
}

/*************************************************************************
 Class : PrimitiveBitString
 Method: getBit
 Params: Number pos
 Return: Boolean isSet
 Description: Returns if the boolean value of the specified bit
   position.
 Exceptions: OutOfRange
*************************************************************************/
function _PrimitiveBitString_getBit( pos )
{
   if ( pos < this.getLength() )
   {
      bitValue=this.data.charAt(pos);
      if(bitValue=="1"){return true; }
      if(bitValue=="0"){return false;}
      throw new CJException.InvalidValue(
      "PrimitiveBitString_getBit - bit position \"" + pos
      + "\" invalid, must be 0 or 1 found \""+bitValue );
   }
   throw new CJException.OutOfRange(
      "PrimitiveBitString_getBit - bit position \"" + pos
      + "\" invalid, range is 0 to " + this.getLength() + "-1" );
}

// get the bit value at the specified position
/*************************************************************************
 Class : PrimitiveBitString
 Method: setBit
 Params: Number pos, Boolean val
 Return: none
 Description: Set's the bit value at the specified position
 Exceptions: OutOfRange & InvalidValue
*************************************************************************/
function _PrimitiveBitString_setBit( pos, val, from )
{
   if ( pos < this.getLength() )
   {
      if( pos > this.getMaxBits() )
      {
         throw new CJException.InvalidValue(
```

```
            "PrimitiveBitString_setBit - at bit position \"" + pos
            + "\" is exceeds maximum bit restriction");
        }
        if(this.hasAllowed())
        {
           if(this.getAllowed().charAt(pos) != "1")
           {
              throw new CJException.InvalidValue(
                "PrimitiveBitString_setBit - at bit position \"" + pos
                + "\" is restricted");
           }
        }
        // if the value is 0, false, "false, or null set to false
        // if it is anything else set to true
        if ( val == 0 || val == false || val=="false" || val == null )
        {
           bitValue = "0";
        }
        else
        {
           bitValue = "1";
        }
        var newString = null;
        // if setting the position of the first bit
        if (pos==0)
        {
           newString =  bitValue+this.data.substr(pos+1);
        }
        //if setting the position of the last bit
        else if (pos==this.getLength())
        {
           newString =  this.data.slice(0, pos)+bitValue;
        }
        else
        {
        //if setting the position of any other bit
          newString =  this.data.slice(0, pos)+bitValue+this.data.substr(pos+1);
        }
        this.setValue( newString, from );
        return newString;
     }
     throw new CJException.OutOfRange(
        "PrimitiveBitString_setBit - bit position \"" + pos
        + "\" invalid, range is 0 to " + this.getLength() + "-1" );
}
```

```
    // access the set definition
    function _PrimitiveBitString_getSetDefinition( )
    {
2370    return this.enumDef;
    }

    // get the element text for the specified bit position.
    function _PrimitiveBitString_getElementText( bit )
2375 {
    return this.enumDef.getElementText( bit );
    }


2380 /*************************************************************************
     Class : PrimitiveBitString
     Method: getMaxBits, hasMaxBits and setMaxBits
     Params: varies
     Return: varies
2385 Description: a restriction for the PrimitiveBitString is the maximum
                  number of bits these methods handle this restriction.
    *************************************************************************/
    function _PrimitiveBitString_getMaxBits()
    {
2390    if ( this.maxbits == null )
          return Number.MAX_VALUE;
        else
          return this.maxbits;
    }
2395
    function _PrimitiveBitString_hasMaxBits()
    {
        return this.maxbits != null;
    }
2400
    function _PrimitiveBitString_setMaxBits( max )
    {
        if ( max < 0 )
        {
2405      throw new CJException.InvalidValue("MaxBits " + max + " must be >= 0" );
        }
        this.maxbits = max;
    }

2410 /*************************************************************************
     Class : PrimitiveBitString
     Method: getAllowed, hasAllowed and setAllowed
```

```
     Params: varies
     Return: varies
2415 Description: a restriction for the PrimitiveBitString is which bits are
            allowed to be set these methods handle this restriction.
    ************************************************************************/
    function _PrimitiveBitString_getAllowed()
    {
2420    return this.allowed;
    }

    function _PrimitiveBitString_hasAllowed()
    {
2425    return this.allowed != null;
    }

    function _PrimitiveBitString_setAllowed( allowed )
    {
2430    this.allowed = allowed;
    }
    /************************************************************************
     Class : PrimitiveBitString
     Method: validateValue
2435 Params: BitString test
     Return: null
     Description: checks the test value against the primitive's restrictions
    ************************************************************************/
    function _PrimitiveBitString_validateValue( test )
2440 {
       if ( test.length > this.getMaxBits() )
       {
          throw new CJException.InvalidValue("BitString of " + test.length +
                " bits exceeds limit of " + this.getMaxBits() + " bits");
2445    }
       // get and save our "allow bit positions" - subset of total
       var allowed = this.getAllowed();

       // go over each bit and check it
2450   for (var i=0; i<test.length; i++ )
       {
          var c = test.charAt(i);

          // ensure it contains only '0' and '1' chars
2455      if ( c != '1' && c != '0' )
          {
             throw new CJException.InvalidValue("PrimitiveBitString: Invalid BitString \""
                                        + test + "\" may contain only '0' and '1'\n");
```

```
        }

        // ensure that only allowed bits, if any, are set
        if ( allowed != null && c == '1' &&
           (i >= allowed.length || allowed.charAt(i) != '1' ))
        {
           throw new CJException.InvalidValue("BitString \"" + test +
               "\" contains disallowed bits, mask is \"" + allowed + "\"\n");
        }

        // ensure that an "on" bit is in our definition (if we have one)
        if ( c == '1' && this.enumDef != null )
        {
           try
           {
             this.enumDef.getElementText(i)
           }
           catch ( ex )
           {
              if(ex instanceof CJException.NotFound)
                 throw new CJException.InvalidValue("BitString \"" + test +
                         "\" contains undefined bit at offset " + i + "\n");
              else
                 throw ex;
           }
        }
    }
}

PrimitiveBitString.prototype = new PrimitiveBase();
PrimitiveBitString.prototype.valueChanged     = _PrimitiveBitString_valueChanged;
PrimitiveBitString.prototype.getLength        = _PrimitiveBitString_getLength;
PrimitiveBitString.prototype.getSetDefinition = _PrimitiveBitString_getSetDefinition;
PrimitiveBitString.prototype.getBit           = _PrimitiveBitString_getBit;
PrimitiveBitString.prototype.setBit           = _PrimitiveBitString_setBit;
PrimitiveBitString.prototype.getElementText   = _PrimitiveBitString_getElementText;
PrimitiveBitString.prototype.maxbits          = null;
PrimitiveBitString.prototype.getMaxBits       = _PrimitiveBitString_getMaxBits;
PrimitiveBitString.prototype.hasMaxBits       = _PrimitiveBitString_hasMaxBits;
PrimitiveBitString.prototype.setMaxBits       = _PrimitiveBitString_setMaxBits;
PrimitiveBitString.prototype.allowed          = null;
PrimitiveBitString.prototype.getAllowed       = _PrimitiveBitString_getAllowed;
PrimitiveBitString.prototype.hasAllowed       = _PrimitiveBitString_hasAllowed;
PrimitiveBitString.prototype.setAllowed       = _PrimitiveBitString_setAllowed;
PrimitiveBitString.prototype.validateValue    = _PrimitiveBitString_validateValue;
```

```
2505


2510 /*****************************************************************
     *****************************************************************

       CLASS: PrimitiveExpr

2515   A PrimitiveExpr is a data value that is the result of an
       arbitrary JavaScript expression.  If is constructed with
       initial data that is the expression to be evaluated, and
       a list of primitive IDs that are the other primitives that
       the expression depends on.  If any of the dependent primitives
2520   are undefined, then the current value is undefined.  Once
       all of the primitives are defined, then the expression may
       and will be evaluated.

       Being writable implies that the *expression* may be redefined.
2525
       It is illegal to invoke setValue()/setMasterValue() on this.
       The master value is always 'null'.  Therefore calling isChanged()
       is rather meaningless.

2530   To change the expression, use setExpression()

       The class's data type is ANY.

     *****************************************************************
2535 *****************************************************************/
     function PrimitiveExpr( id, writable,  initExpr, depList )
     {
        this.expression = initExpr;
        this.dependents = null;
2540    this.setDependents( depList );

        this.PrimitiveBase( id, writable,  null );

        this.evaluate();
2545 }

     /*****************************************************************
      Class : PrimitiveExpr
      Method: setExpression
2550  Params: String newExpr    new JavaScript expression
```

```
            Array depList     Stringarray of primitive IDs (or null if none)
     Return: none
     Description: Sets a new expression and dependency list
     Exceptions: none
2555 ************************************************************************/
     function _PrimitiveExpr_setExpression( newExpr, depList )
     {
        if ( this.isWritable() )
        {
2560       this.expression = newExpr;
           this.setDependents( depList );
           this.evaluate();
        }
        else
2565    {
           throw new CJException.ReadOnly(
              "Cannot set expression for primitive \"" + this.id + "\"" );
        }
     }
2570
     /************************************************************************
      Class : PrimitiveExpr
      Method: valueChanged
      Params: none
2575  Return: none
      Description: The valueChanged callback is invoked whenever a dependent
        primitive is changed.  The PrimitiveExpr primitive will register
        itself as a listener of the dependent primitives.
      Exceptions: none
2580 ************************************************************************/
     function _PrimitiveExpr_valueChanged( )
     {
        this.evaluate();
     }
2585
     /************************************************************************
      Class : PrimitiveExpr
      Method: evaluate
      Params: none
2590  Return: none
      Description: Evaluates the expression.  If any of the dependent
        primtives are undefined, then it is assumed that the expression
        cannot be properly validated and therefore the primitive will
        be 'null'.  Otherwise the result of the expression is stored
2595    in the current value.
```

```
     Exceptions: none
    ************************************************************************/
    function _PrimitiveExpr_evaluate( )
2600 {
       var newVal = null;
       var isUndefined = false;

       // check to see if any dependents are undefined
2605   if ( this.dependents != null )
       {
          for ( var i=0; i<this.dependents.length; i++ )
          {
             if ( this.dependents[i].isUndefined() )
2610         {
                isUndefined = true;
                break;
             }
          }
2615   }
       // evaluate only if not undefined
       if ( ! isUndefined )
       {
          try
2620      {
             newVal = eval( this.expression );
          }
          catch  ( e )
          {
2625         throw new CJException.InvalidValue(
                "PrimitiveExpr - cannot evaluate expression for primitive \"" +
                this.getId() + "\"\n" + e.toString() );
          }
       }
2630
       // test for change
       if ( this.data != newVal )
       {
          // update value and notify
2635      this.data = newVal;
          this.notifyListeners( this );
          PrimitiveBase.notifyUIListeners( this, this );
       }
    }
2640
    /************************************************************************
     Class : PrimitiveExpr
```

```
     Method: setDependents (PRIVATE HELPER)
     Params: Array depList     Stringarray of primitive IDs (or null if none)
2645 Return: none
     Description: Unregisters from previously listened to primitives, then
       finds all of the new primitives and registers as a listener to them.
     Exceptions: none
    ************************************************************************/
2650 function _PrimitiveExpr_setDependents( depList )
     {
        if ( this.dependents != null )
        {
           for ( var i=0; i<this.dependents.length; i++ )
2655       {
              this.dependents[i].removeListener( this );
           }
           this.dependents = null;
        }
2660
        if ( depList != null && depList.length > 0 )
        {
           this.dependents = new Array();
           for ( var i=0; i<depList.length; i++ )
2665       {
              this.dependents[i] = PrimitiveBase.findPrimitive( depList[i] );
              this.dependents[i].addListener( this );
           }
        }
2670 }

     // INVALID FUNCTION - NOT ALLOWED - throw exception
     function _PrimitiveExpr_setValue()
     {
2675    throw new CJException.InvalidValue(
           "PrimitiveExpr - setValue() is not allowed for primitive \"" +
              this.getId() + "\"" );
     }

2680 // INVALID FUNCTION - NOT ALLOWED - throw exception
     function _PrimitiveExpr_setMasterValue()
     {
        throw new CJException.InvalidValue(
           "PrimitiveExpr - setMasterValue() is not allowed for primitive \"" +
2685          this.getId() + "\"" );
     }
```

```
PrimitiveExpr.prototype = new PrimitiveBase();
PrimitiveExpr.prototype.setExpression   = _PrimitiveExpr_setExpression;
PrimitiveExpr.prototype.evaluate        = _PrimitiveExpr_evaluate;
PrimitiveExpr.prototype.valueChanged        = _PrimitiveExpr_valueChanged;
PrimitiveExpr.prototype.setDependents   = _PrimitiveExpr_setDependents;
PrimitiveExpr.prototype.setValue        = _PrimitiveExpr_setValue;
PrimitiveExpr.prototype.setMasterValue  = _PrimitiveExpr_setMasterValue;


/*************************************************************************
**************************************************************************

   CLASS: PrimitiveTable

   PrimitiveTable consists of an array of PrimTableRows.  PrimTableRows
   have an id and an array of PrimTableCells.  A PrimTableCell has a
   display string and a value string.  The display string is used to
   represent the cell in a control and the value string is used when
   communicating with core.

    add and removeRow are used to manipulate the table.

**************************************************************************
**************************************************************************/

function PrimTableCell(value)
{
   this.value = value;
}

function _PrimTableCell_GetCopy()
{
   return (new PrimTableCell(this.value));
}

function _PrimTableCell_ToString()
{
   var text;
   var ch;

   text = "";

   for (var i = 0; i < this.value.length; i++)
```

```
2735    {
           ch = this.value.charAt(i);

           if ((ch == PrimitiveTable.FORMAT_CELLDATA_SEP) ||
               (ch == PrimitiveTable.FORMAT_CELLDATA_END) ||
2740           (ch == PrimitiveTable.FORMAT_ESCAPE_CHAR))
           {
              text += PrimitiveTable.FORMAT_ESCAPE_CHAR;
           }

2745       text += ch;
        }

        return (text);
     }
2750

     PrimTableCell.prototype.getCopy = _PrimTableCell_GetCopy;
     PrimTableCell.prototype.toString = _PrimTableCell_ToString;

2755
     // note that the cells param is an array of PrimTableCells
     function PrimTableRow(id, setDefId, cells, userAdded)
     {
        this.id = id;
2760    this.setDefId = setDefId;
        this.cells = cells;
        this.userAdded = userAdded;
        this.changed = false;
        this.deleted = false;
2765 }


     function _PrimTableRow_GetCopy()
     {
2770    var newCells;
        var newRow;

        newCells = new Array();

2775    for (var i = 0; i < this.cells.length; i++)
        {
           newCells[i] = null;

           if (this.cells[i] != null)
2780       {
```

```
            newCells[i] = this.cells[i].getCopy();
         }
      }

2785  newRow = new PrimTableRow(this.id, this.setDefId, newCells, this.userAdded);
      newRow.changed = this.changed;
      newRow.deleted = this.deleted;

      return (newRow);
2790 }


   PrimTableRow.prototype.getCopy = _PrimTableRow_GetCopy;

2795
   /*************************************************************************
    Class : PrimitiveTable
    Method: PrimitiveTable (constructor)
    Params: id - unique id for primitive
2800        writable - if false add/removeRow will fail
            initData - Array of PrimTableRows
            local - true if the primitive is local to a page and should not
                    be submitted after a change
    Description: Note that initData is not a string, it is a literal array
2805        of objects.
   *************************************************************************/

   function PrimitiveTable(id, writable, initData, numberOfColumns, local)
   {
2810  this.PrimitiveBase(id, writable, initData);

      this.numberOfColumns = numberOfColumns;

      if (local != null)
2815  {
         this.setPageLocal(local);
      }

   }
2820
   /*************************************************************************
    Class : PrimitiveTable
    Method: addRow
    Params: Object from
2825 Return: Nothing.
    Description: Adds a row to the table.
```

```
    ***************************************************************************/

    function _PrimitiveTable_AddRow(from, rowId, setDefId, cellsData)
2830 {
       var limit = this.data.length;
       var cellsArray;

       if (rowId == null)
2835   {
          rowId = PrimitiveTable.ID_NEW;
       }

       if (setDefId == null)
2840   {
          setDefId = "0";
       }

       if (cellsData == null)
2845   {
          cellsArray = new Array();

          for (var index = 0; index < this.numberOfColumns; index++)
          {
2850         cellsArray[index] = new PrimTableCell("");
          }
       }
       else
       {
2855      cellsArray = cellsData;
       }

       this.data[limit] = new PrimTableRow(rowId, setDefId, cellsArray, true);
       this.data[limit].changed = true;
2860
       this.changed = true;

       // Notify the listeners.
       this.notifyListeners(from);
2865
       if (!this.pageLocal)
       {
          PrimitiveBase.notifyUIListeners(this, from);
       }
2870 }

    /***************************************************************************
```

```
     Class : PrimitiveTable
     Method: clearChanged
2875 Params:
     Return:
     Description: Called to set the changed flag for the table to false.
    ***************************************************************************/

2880 function _PrimitiveTable_ClearChanged()
     {
        var newData;
        var index;

2885    this.changed = false;

        // Remove the deleted rows.
        newData = new Array();
        index = 0;
2890
        for (var i = 0; i < this.data.length; i++)
        {
           if (this.data[i].deleted != true)
           {
2895          newData[index] = this.data[i];
              newData[index].deleted = false;
              newData[index].changed = false;
              newData[index].userAdded = false;
              this.data[i] = null;
2900
              ++index;
           }
        }

2905    this.data = newData;

        // Set the original data to this one too.
        this.origData = new Array();

2910    for (var i = 0; i < this.data.length; i++)
        {
           this.origData[i] = this.data[i].getCopy();
        }

2915 }

     /***************************************************************************
      Class : PrimitiveTable
```

```
     Method: deleteRow
2920  Params: Number rowIndex
             Object from
     Return: Nothing.
     Description: Deletes a row from the table.
    ***********************************************************************/
2925
    function _PrimitiveTable_DeleteRow(rowIndex, from)
    {
       var realIndex;

2930    realIndex = this.getRealIndex(rowIndex);

       // THROW EXCEPTION HERE
       CJAssert(realIndex != null, "Requested unknown row");

2935    this.data[realIndex].deleted = true;
       this.data[realIndex].changed = true;

       this.changed = true;

2940    // Notify the listeners.
       this.notifyListeners(from);

       if (!this.pageLocal)
       {
2945       PrimitiveBase.notifyUIListeners(this, from);
       }

    }

2950 /***********************************************************************
     Class : PrimitiveTable
     Method: fromString
     Params: String text
     Return: Array of PrimTableRows.
2955 Description: Converts the string given into a PrimitiveTable form.
    ***********************************************************************/

    function _PrimitiveTable_FromString(text)
    {
2960    var data = null;
       var ch;
       var state;
       var rowId;
       var setDefId;
```

```
2965    var value;
        var ignoreCode;
        var cellsArray;
        var userAdded = false;

2970    var STATE_GETTING_ROW = 0;
        var STATE_GETTING_ROWID = 1;
        var STATE_GETTING_SETDEFID = 2;
        var STATE_GETTING_VALUE = 3;

2975    // Check for the special code to add a new row.
        if (text.indexOf(PrimitiveTable.ID_NEW) == 0)
        {
           // Get a copy of the current data.
           data = new Array();
2980
           for (var i = 0; i < this.data.length; i++)
           {
              data[i] = this.data[i].getCopy();
           }
2985
           text = text.substring(PrimitiveTable.ID_NEW.length);
           userAdded = true;
        }

2990    if (data == null)
        {
           data = new Array();
        }

2995    state = STATE_GETTING_ROW;
        ignoreCode = false;

        for (var i = 0; i < text.length; i++)
        {
3000       ch = text.charAt(i);

           switch (state)
           {
              case STATE_GETTING_ROW:
3005
                 if (ch != PrimitiveTable.FORMAT_ROWINDICATOR)
                 {
                    throw new CJException.InvalidClass("_PrimitiveTable_FromString - input text \"" + text + "\"
expecting to find a '*' t
                 }
```

```
3010
                rowId = "";
                setDefId = "";
                state = STATE_GETTING_ROWID;
                break;
3015
            case STATE_GETTING_ROWID:

                if (ch == PrimitiveTable.FORMAT_ROWID_SEP)
                {
3020                state = STATE_GETTING_SETDEFID;
                }
                else
                {
                    rowId += ch;
3025            }
                break;

            case STATE_GETTING_SETDEFID:

3030            if (ch == PrimitiveTable.FORMAT_CELLDATA_START)
                {
                    cellsArray = new Array();

                    data[data.length] = new PrimTableRow(rowId, setDefId, cellsArray, userAdded);
3035
                    value = "";

                    state = STATE_GETTING_VALUE;
                }
3040            else
                {
                    setDefId += ch;
                }
                break;
3045
            case STATE_GETTING_VALUE:

                if (ch == PrimitiveTable.FORMAT_ESCAPE_CHAR)
                {
3050                ignoreCode = true;
                }
                else
                {
                    if (ignoreCode == false)
3055                {
```

```
                    if (ch == PrimitiveTable.FORMAT_CELLDATA_END)
                    {
                       cellsArray[cellsArray.length] = new PrimTableCell(value);
3060                   value = "";
                       if (i < (text.length - 1))
                       {
                          if (text.charAt(i + 1) == PrimitiveTable.FORMAT_ROWINDICATOR)
3065                      {
                             state = STATE_GETTING_ROW;
                          }
                          else
                          {
3070                         state = STATE_GETTING_VALUE;
                          }
                       }
                    }
                    else
3075                {
                       value += ch;
                    }
                 }
                 else
3080             {
                    value += ch;
                    ignoreCode = false;
                 }
3085          }
              break;
           default:
3090
              throw new CJException.InvalidClass("_PrimitiveTable_FromString - invalid state");
              break;
        }
3095
     }
     return (data);
  }
3100
  /*************************************************************************
```

```
     Class : PrimitiveTable
     Method: getColumn
     Params: Number          rowIndex.
3105         Number          column.
     Return: PrimTableCell   columnData.
     Description: Sets the column to the stuff specified.
    ************************************************************************/

3110 function _PrimitiveTable_GetColumn(rowIndex, column)
    {
       realIndex = this.getRealIndex(rowIndex);

       CJAssert(realIndex != null, "Requested unknown row");
3115   CJAssert(this.data[realIndex].cells[column] != null, "Requested unknown column");

       return (this.data[realIndex].cells[column]);
    }

3120 /************************************************************************
     Class : PrimitiveTable
     Method: getNumberOfRows
     Params:
     Return: PrimTableRow
3125 Description: Returns the number of rows in the table.
    ************************************************************************/

    function _PrimitiveTable_GetNumberOfRows()
    {
3130   var count = 0;

       if (this.data == null)
       {
          return (0);
3135   }

       for (var i = 0; i < this.data.length; i++)
       {
          if (this.data[i].deleted != true)
3140       {
             ++count;
          }
       }

3145   return (count);
    }
```

```
    /*************************************************************************
     Class : PrimitiveTable
3150 Method: getRow
     Params: rowId - Row to return.
     Return: PrimTableRow
     Description: Returns a row specified by the index given.
    *************************************************************************/
3155
    function _PrimitiveTable_GetRow(rowIndex)
    {
       var realIndex;

3160   realIndex = this.getRealIndex(rowIndex);

       // THROW EXCEPTION HERE
       CJAssert(realIndex != null, "Requested unknown row");

3165   return (this.data[realIndex]);

    }

    /*************************************************************************
3170 Class : PrimitiveTable
     Method: getRowId
     Params: rowIndex - Row to return.
     Return: id
     Description: Returns the row id of a row index.
3175 *************************************************************************/

    function _PrimitiveTable_GetRowId(rowIndex)
    {
       var realIndex;
3180
       realIndex = this.getRealIndex(rowIndex);

       // THROW EXCEPTION HERE
       CJAssert(realIndex != null, "Requested unknown row");
3185
       return (this.data[realIndex].id);
    }


3190
    /*************************************************************************
     Class : PrimitiveTable
     Method: getSetDefId
```

```
     Params: rowIndex - Row to return.
3195  Return: id
     Description: Returns the row id of a row index.
    ***********************************************************************/

    function _PrimitiveTable_GetSetDefId(rowIndex)
3200 {
       var realIndex;

       realIndex = this.getRealIndex(rowIndex);

3205   // THROW EXCEPTION HERE
       CJAssert(realIndex != null, "Requested unknown row");

       return (this.data[realIndex].setDefId);
    }
3210


    /***********************************************************************
     Class : PrimitiveTable
3215 Method: GetUndoAddString
     Params:
     Return: String
     Description: Returns the string format that should be sent to core to undo
                  any new rows that were added by the user.
3220 ***********************************************************************/

    function _PrimitiveTable_GetUndoAddString()
    {
       var text = "";
3225
       for (var index = 0; index < this.data.length; index++)
       {
          if (this.data[index].userAdded == true)
          {
3230         text += PrimitiveTable.FORMAT_ROWINDICATOR;
             text += PrimitiveTable.ID_DELETED + this.data[index].id;
             text += PrimitiveTable.FORMAT_ROWID_SEP;
             text += this.data[index].setDefId;
             text += PrimitiveTable.FORMAT_CELLDATA_START;
3235
             for (var cell = 0; cell < this.data[index].cells.length; cell++)
             {
                text += this.data[index].cells[cell].toString();
                text += PrimitiveTable.FORMAT_CELLDATA_END;
```

```
            }
         }
      }

      return (text);
   }

   /**********************************************************************
    Class : PrimitiveTable
    Method: ifUserAdded
    Params:
    Return: Boolean
    Description: Returns a true if the user has added any rows before any changes
                 are posted to core.
   **********************************************************************/

   function _PrimitiveTable_IfUserAdded()
   {
      for (var i = 0; i < this.data.length; i++)
      {
         if (this.data[i].userAdded == true)
         {
            return (true);
         }
      }

      return (false);
   }

   /**********************************************************************
    Class : PrimitiveTable
    Method: setColumn
    Params: Number         rowIndex.
            Number         column.
            PrimTableCell  columnData.
            Boolean        notifyUIListeners.
    Return:
    Description: Sets the column to the stuff specified.
   **********************************************************************/

   function _PrimitiveTable_SetColumn(from, rowIndex, column, columnData, notifyUIListeners)
   {
      realIndex = this.getRealIndex(rowIndex);
```

```
        CJAssert(realIndex != null, "Requested unknown row");
        CJAssert(this.data[realIndex].cells[column] != null, "Requested unknown column");

3290    this.data[realIndex].cells[column] = columnData;
        this.data[realIndex].changed = true;
        this.data[realIndex].deleted = false;

        this.changed = true;
3295
        // Notify the listeners.
        this.notifyListeners( from );

        if ((!this.pageLocal) && (notifyUIListeners))
3300    {
           PrimitiveBase.notifyUIListeners(this, null);
        }

    }
3305

    /**************************************************************************
     Class : PrimitiveBase
     Method: setMasterValue
3310 Params: object newValue
     Return: none
     Description: Sets the master value of the primitive unconditionally
       and the current value only if never set by the user.
       It then notifies all listeners of the change.
3315
       Overrides the base class.
    **************************************************************************/

    function _PrimitiveTable_setMasterValue(newData)
3320 {
        var changed = false;

        if (newData.indexOf(PrimitiveTable.ID_NEW) == 0)
        {
3325       changed = true;
        }

        newData = this.fromString(newData);

3330    if (changed == true)
        {
```

```
         this.data = newData;

         this.changed = true;
3335     this.notifyListeners();
      }
      else
      {
         this.origData = newData;
3340
         if (!this.changed)
         {
            this.resetData();
            this.changed = changed;
3345        this.notifyListeners();
         }
      }

      if ((this.changed == true) && (!this.pageLocal))
3350  {
         PrimitiveBase.notifyUIListeners(this, null);
      }
   }

3355

   /**************************************************************************
    Class : PrimitiveTable
    Method: toString
3360  Params:
    Return: String
    Description: PrimitiveTable in String form:
                 "*id,setDefId:value;value;*id:'....etc'"
                 *id indicates the begining of a new row and id is
3365             the id of the row.
                 It will only include the rows that have changed.
   **************************************************************************/

   function _PrimitiveTable_ToString()
3370 {
      var text = "";

      for (var index = 0; index < this.data.length; index++)
      {
3375     if (this.data[index].changed == true)
         {
            text += PrimitiveTable.FORMAT_ROWINDICATOR;
```

```
            if (this.data[index].deleted == true)
            {
3380           text += PrimitiveTable.ID_DELETED;
            }
            text += this.data[index].id;
            text += PrimitiveTable.FORMAT_ROWID_SEP;
            text += this.data[index].setDefId;
3385        text += PrimitiveTable.FORMAT_CELLDATA_START;

            for (var cell = 0; cell < this.data[index].cells.length; cell++)
            {
               text += this.data[index].cells[cell].toString();
3390           text += PrimitiveTable.FORMAT_CELLDATA_END;
            }
         }
      }

3395
      return (text);

   }

3400
   function _PrimitiveTable_resetData()
   {
      if (this.origData == null)
      {
3405     this.data = null;
      }
      else
      {
         this.data = new Array();
3410
         for (var i = 0; i < this.origData.length; i++)
         {
            this.data[i] = this.origData[i].getCopy();
         }
3415  }
   }

   /*************************************************************************
3420  Class : PrimitiveTable
      Method: getRealIndex
      Params:
      Return: Number or null if not valid.
```

```
     Description: Gets the real index in the data array.
3425 ************************************************************************/

     function _PrimitiveTable_GetRealIndex(rowIndex)
     {
        var realIndex;
3430    var fakeIndex = -1;

        for (realIndex = 0; realIndex < this.data.length; realIndex++)
        {
           if (this.data[realIndex].deleted != true)
3435       {
              ++fakeIndex;
           }

           if (rowIndex == fakeIndex)
3440       {
              return (realIndex);
           }
        }

3445    return (null);
     }


3450 PrimitiveTable.prototype                  = new PrimitiveBase();
     PrimitiveTable.prototype.addRow           = _PrimitiveTable_AddRow;
     PrimitiveTable.prototype.clearChanged     = _PrimitiveTable_ClearChanged;
     PrimitiveTable.prototype.deleteRow        = _PrimitiveTable_DeleteRow;
     PrimitiveTable.prototype.fromString       = _PrimitiveTable_FromString;
3455 PrimitiveTable.prototype.getColumn        = _PrimitiveTable_GetColumn;
     PrimitiveTable.prototype.getNumberOfRows  = _PrimitiveTable_GetNumberOfRows;
     PrimitiveTable.prototype.getRow           = _PrimitiveTable_GetRow;
     PrimitiveTable.prototype.getRowId         = _PrimitiveTable_GetRowId;
     PrimitiveTable.prototype.getSetDefId      = _PrimitiveTable_GetSetDefId;
3460 PrimitiveTable.prototype.getUndoAddString = _PrimitiveTable_GetUndoAddString;
     PrimitiveTable.prototype.ifUserAdded      = _PrimitiveTable_IfUserAdded;
     PrimitiveTable.prototype.setColumn        = _PrimitiveTable_SetColumn;
     PrimitiveTable.prototype.setMasterValue   = _PrimitiveTable_setMasterValue;
     PrimitiveTable.prototype.toString         = _PrimitiveTable_ToString;
3465 PrimitiveTable.prototype.resetData         = _PrimitiveTable_resetData;

     PrimitiveTable.prototype.getRealIndex     = _PrimitiveTable_GetRealIndex;
```

```
3470 PrimitiveTable.FORMAT_ROWINDICATOR = '*';
     PrimitiveTable.FORMAT_ROWID_SEP = ',';
     PrimitiveTable.FORMAT_CELLDATA_START = ':';
     PrimitiveTable.FORMAT_CELLDATA_SEP = ',';
     PrimitiveTable.FORMAT_CELLDATA_END = ';';
3475 PrimitiveTable.FORMAT_ESCAPE_CHAR = '\\';

     PrimitiveTable.CODE_SELECTED = "$ROW_SELECTED$";

     PrimitiveTable.ID_DELETED = "$ROW_DELETED$";
3480 PrimitiveTable.ID_NEW = "$ROW_ADDNEW$";
```

Exhibit 9

```
/*****************************************************************************
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
        This document contains confidential/proprietary information.
*****************************************************************************

@(#)WidgetTextInput.js
Initial Author: Jide Omisore

Description: Text Input Box widget. This will be used for many controls.

Dependencies:
  /util/browserapi.js
  /util/cjelementhelper.js
  /util/compat.js
  /controls/CJEvent.js
  /controls/exception.js
  /controls/iconhelper.js

$Log: /green/webroot/common/controls/WidgetTextInput.js $
*
* 29    12/01/99 2:25p Sappling
* changed setSize to be based on font size
*
* 28    11/30/99 11:29a Ddavidson
* fixed focusText bug where is always triggered a change
*
* 27    11/29/99 3:14p Ddavidson
* added focus() methods to focus to (first) text field
*
* 26    11/18/99 3:17p Ddavidson
* added focusText member to deal with IE not calling onChange when
* clicking outside of the IFRAME.
*
* 25    9/27/99 9:34a Jomisore
* Fixed a bug in the setEditable method.
*
* 24    9/22/99 3:40p Jomisore
* Made changes to the style stuff so that things run faster.
*
* 23    9/22/99 12:01p Sappling
* This has been changed to partially implement the new styles
*
* 22    9/09/99 9:46a Jomisore
* Added the "invalidateValue" method
*
* 21    9/08/99 4:35p Jomisore
* Fixed some comments, cleaned up code for code review.
*
* 20    8/16/99 2:23p Jomisore
```

```
     * Changed it so that when the widget control is disabled it doesn't set
     * the disabled property of the html element to true.
     *
     * 19    8/13/99 12:31p Jomisore
55   * Added workaround code for the IE crash bug.
     *
     * 18    8/10/99 9:39a Jomisore
     * Fixed a problem with the master value in the text input box not being
     * reset in some cases.
60   *
     * 17    8/09/99 9:17a Jomisore
     * Added a TextDisplayWidget.
     *
     * 16    8/06/99 11:45a Jomisore
65   * Changed it so that if size is specified as 0 in the constructor, it
     * will automaitcally adjust the size of the input box to be the number of
     * characters+1.
     *
     * 15    8/04/99 6:02p Jomisore
70   * Added new style management.
     *
     * 14    8/04/99 2:06p Jomisore
     * Fixed setValue so that it can accept nulls.
     *
75   * 13    8/04/99 11:49a Jomisore
     * Passed "this" as a parameter to the widget callbacks.
     *
     * 12    8/04/99 10:34a Jomisore
     * Added some more comments.
80   *
     * 11    8/03/99 4:49p Jomisore
     * Added category for distinguishing between styles.
     *
     * 10    8/02/99 2:02p Jomisore
85   * Changed the names of the owner callbacks used for handling events.
     * Swapped the priority of the styles.
     *
     * 9     7/29/99 5:25p Jomisore
     * Line fix.
90   *
     * 8     7/29/99 10:09a Jomisore
     * Fixed the tabindex to -1 if the widget is not writable or is disabled.
     *
     * 7     7/28/99 12:40p Jomisore
95   * Fixed the style adapter that is set when the widget is made not
     * writable.
     *
     * 6     7/28/99 10:46a Jomisore
     * Added the affectStyleById method.
100  *
```

```
     * 5     7/28/99 9:15a Jomisore
     * Fixed the CJAsserts and cleaned up some code.
     *
     * 4     7/23/99 6:30p Jomisore
105  * Added comments.
     *
     * 3     7/23/99 6:26p Jomisore
     * Fixed it to use the style manager and added more functionality to it.
     *
110  * 1     7/16/99 1:03p Jomisore
     * Renamed from TextInputBox.
     *
     * 3     7/15/99 6:32p Jomisore
     * Oops i broke the keypress validation.
115  *
     * 1     7/15/99 2:53p Jomisore
     * Text input box widget and validation.
     *
     * 1     07/14/99 3:28p jomisore
120  * Initial implementation

    ******************************************************************************/


125
    /******************************************************************************
     Class : WidgetTextInput

             Public methods:
130
                applyOnBlurStyle
                applyOnFocusStyle
                getDisabled
                getEditable
135             getValue
                isValueValid
                setDisabled
                setEditable
                setListener
140             setSize
                setStyleState
                setValue
                toElement

145 ******************************************************************************/

    /******************************************************************************
     Class : WidgetTextInput
     Method: WidgetTextInput   Constructor
150  Params: Boolean editable                    -  Flags if this is editable or not.
```

```
            Boolean disabled                    -  Flags if this is disabled or not.
            Number tabIndex                     -  Tab index of the field.
            ValidationManager validationRule    -  Validation rule(s) to apply.
            Boolean validateOnKeyPress          -  Whether to validate each
155                                                 key
            String category                     -  This is a string identifying
                                                   what category the widget belongs
                                                   to. This name will also be used
                                                   as a prefix in the name used for
160                                                looking up styles.
            Number size                         -  Size of the text field in
                                                   characters. If the alue it 0
                                                   then the widget will
                                                   automatically adjust to the size
165                                                of the input value.
            Number maxLength                    -  Maximum number of characters
                                                   that can be entered.
            Anything value                      -  Initial value of the field.
     Return:
170  Description:  Our text widget.
    ********************************************************************************/
    function WidgetTextInput(editable, disabled, tabIndex, validationRule,
                             validateOnKeyPress, category, size, maxLength, value)
    {
175    // Set the defaults for parameters that are not required.
       this.editable = true;
       this.disabled = false;
       this.validationRule = null;
       this.validateOnKeyPress = false;
180    this.tabIndex = null;
       this.focusText = null;  // for IE bug - see OnBlur & OnChange

       // Set the values that have been defined.
       if (editable != null)
185    {
          this.editable = editable;
       }

       if (disabled != null)
190    {
          this.disabled = disabled;
       }

       if (validationRule)
195    {
          this.validationRule = validationRule;
       }

       if (validateOnKeyPress)
200    {
```

```
        this.validateOnKeyPress = validateOnKeyPress;
    }

    // Other fields.
    this.listenerObj = null;

    // Create the element.
    this.element = document.createElement("INPUT");

    // Set some attributes.
    this.element.type = "text";

    if (maxLength)
    {
        this.element.maxLength = maxLength;
    }

    if (tabIndex != null)
    {
        this.tabIndex = this.element.tabIndex = tabIndex;
    }

    this.element.cjowner = this;

    CJElementHelper.registerElement(this.element, "cjowner");

    // Event handlers.
    this.element.onchange = _WidgetTextInput_InputElement_OnChange;
    this.element.onblur = _WidgetTextInput_InputElement_OnBlur;
    this.element.onfocus = _WidgetTextInput_InputElement_OnFocus;

    if (this.validateOnKeyPress == true)
    {
        this.element.onkeypress = _WidgetTextInput_InputElement_OnKeyPress;
    }

    // Styles.
    // If category is not defined then use the default.
    if (!category)
    {
        category = WidgetTextInput.CATEGORY;
    }

    this.category = category;

    CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_BASE, true);

    // Handle editable, dsisabled and so on.....
    this.setEditable(this.editable, true);
    this.setDisabled(this.disabled, true);
```

```
      this.element.value = "";

      if (value != null)
      {
         this.element.value = value;
      }

      this.setSize(size);

      // Check for validation.
      this.validateValue();
   }


   /**************************************************************************
    Constants
   **************************************************************************/


   // Category used if one is not specified by the user.
   WidgetTextInput.CATEGORY = "WidgetTextInput";


   /**************************************************************************
      Basic style extension.
      "-writable"    = Writable style for the control (Main container).
      "-invalid"     = Invalid style for the control (Main container).
      "-focus"       = Focus style for the control (Main container).
      "-disabled"    = Disabled style for the control (Main container).
   **************************************************************************/


   // Extensions for the styles.
   WidgetTextInput.STYLE_EXT_BASE = "-base";
   WidgetTextInput.STYLE_EXT_WRITABLE = "-writable";
   WidgetTextInput.STYLE_EXT_INVALID = "-invalid";
   WidgetTextInput.STYLE_EXT_FOCUS = "-focus";
   WidgetTextInput.STYLE_EXT_DISABLED = "-disabled";


   /**************************************************************************
    Public Methods definitions
   **************************************************************************/
```

```
/*****************************************************************************
 Class : WidgetTextInput
 Method: applyOnBlurStyle
 Params:
 Return:
 Description:  Applies the on blur style to the element.
*****************************************************************************/

function _WidgetTextInput_ApplyOnBlurStyle()
{
   CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_FOCUS, false);
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: applyOnFocusStyle
 Params:
 Return:
 Description:  Applies the on focus style to the element.
*****************************************************************************/


function _WidgetTextInput_ApplyOnFocusStyle()
{
   CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_FOCUS, true);
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: getDisabled
 Params:
 Return: Boolean true or false.
 Description:  Returns the diabled attribute.
*****************************************************************************/


function _WidgetTextInput_GetDisabled()
{
   return (this.disabled);
}
```

```
/*****************************************************************************
 Class : WidgetTextInput
 Method: getEditable
 Params:
 Return: Boolean true or false.
 Description:  Returns the editable attribute.
*****************************************************************************/


function _WidgetTextInput_GetEditable()
{
   return (this.editable);
}



/*****************************************************************************
 Class : WidgetTextInput
 Method: getValue
 Params:
 Return: Any value;
 Description:  Returns this.element.value.
*****************************************************************************/



function _WidgetTextInput_GetValue()
{
   return (this.element.value);
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: invalidateValue
 Params:
 Return:
 Description:  Sets the invalid value flag. It also sets invalid value style.
*****************************************************************************/


function _WidgetTextInput_InvalidateValue()
{
   if (this.valueIsValid == true)
   {
```

```
        this.valueIsValid = false;

        CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_INVALID, true);
    }
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: isInputValid
 Params:
 Return: Boolean true or false.
 Description:  Returns the validation value that is current known.
*****************************************************************************/


function _WidgetTextInput_IsValueValid()
{
    return (this.valueIsValid);
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: setDisabled
 Params: Boolean disabled  -   true or false.
         Boolean force     -  true or false whether to alsways set the state
                              whether the state is changing or not.
 Return:
 Description:  Sets the disabled attribute.
*****************************************************************************/


function _WidgetTextInput_SetDisabled(disabled, force)
{
    // Only do something if it is changing.
    if ((this.disabled != disabled) || (force == true))
    {
        this.disabled = disabled;

        CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_DISABLED, this.disabled);

        // Check the tabbibg index to make sure that the user will not be
        // able to tab to it if it is not editable.
        if (disabled == true)
        {
```

```
            this.element.tabIndex = -1;
         }
         else
         {
455         if ((this.tabIndex != null) && (this.editable == true))
            {
               this.element.tabIndex = this.tabIndex;
            }
         }
460   }
   }


465 /*****************************************************************************
    Class : WidgetTextInput
    Method: setEditable
    Params: Boolean editable  -  true or false.
            Boolean force     -  true or false whether to alsways set the state
470                              whether the state is changing or not.
    Return:
    Description:  Sets the editable attribute.
   *****************************************************************************/

475

   function _WidgetTextInput_SetEditable(editable, force)
   {
      // Only do something if it is changing.
480   if ((this.editable != editable) || (force == true))
      {
         this.editable = editable;
         this.element.readOnly = !this.editable;

485      CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_WRITABLE, this.editable);

         // Check the tabbibg index to make sure that the user will not be
         // able to tab to it if it is not editable.
         if (editable == false)
490      {
            this.element.tabIndex = -1;
         }
         else
         {
495         if ((this.tabIndex != null) && (this.disabled == false))
            {
               this.element.tabIndex = this.tabIndex;
            }
         }
500   }
```

```
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: setListener
 Params: Function listenerObj -  Listener object.
 Return:
 Description:  Sets the listener object for event callbacks.
               The listener object should have any of the following
               methods defined:

               onTextInputChange
               onTextInputBlur
               onTextInputFocus
*****************************************************************************/


function _WidgetTextInput_SetListener(listenerObj)
{
   this.listenerObj = listenerObj;
}


/*****************************************************************************
 Class : WidgetTextInput
 Method: setSize
 Params: Number size          -  Size to set the input field to in characters.
 Return:
 Description:  Sets the size of the input field. If it is 0 then it will set it
               to fit the current value of the field.
*****************************************************************************/


function _WidgetTextInput_SetSize(size)
{
   if (size == null)    // default is auto size
   {
      size = 0;
   }

   this.size = size;

   if (this.size)
   {
      //this.element.size = this.size;
```

```
         this.element.style.width = ((this.size +1)*.6)+"em";
      }
      else
      {
555      if (this.size == 0)
         {
            if (this.element.value != null)
            {
               //this.element.size = this.element.value.toString().length + 1;
560            this.element.style.width = ((this.element.value.toString().length+1) *.6) +"em";
            }
         }
      }
   }
565


   /*****************************************************************************
    Class : WidgetTextInput
570 Method: setStyleState
    Params: String styleClass    -  Class of the style.
            Boolean state        -  state to set it to.
    Return:
    Description:  Sets the style state. The style must exist.
575 *****************************************************************************/



   function _WidgetTextInput_SetStyleState(styleClass, state)
580 {
      CJSetClass(this.element, styleClass, state);
   }


585
   /*****************************************************************************
    Class : WidgetTextInput
    Method: setValue
    Params: Any value.
590 Return: Any value;
    Description:  Sets the value of the field.
   *****************************************************************************/


595
   function _WidgetTextInput_SetValue(value)
   {
      if (value == null)
      {
600      value = "";
```

```
      }

      this.element.value = value;

      // Check validation.
      this.validateValue();

      // Check if the size of the element has to be adjusted.
      if (this.size == 0)
      {
         this.setSize(this.size);
      }
   }


   /*****************************************************************************
    Class : WidgetTextInput
    Method: toElement
    Params:
    Return: HTMLElement
    Description:  Returns the main element associated with this widget.
   *****************************************************************************/


   function _WidgetTextInput_ToElement()
   {
      return (this.element);
   }


   /*****************************************************************************
    Class : WidgetTextInput
    Method: focus
    Params: none
    Return: none
    Description:  Gives the text input element the focus
   *****************************************************************************/
   function _WidgetTextInput_Focus()
   {
      this.element.focus();
   }


   /*****************************************************************************
    Private Methods definitions
   *****************************************************************************/
```

```
/*****************************************************************************
 Class : WidgetTextInput
 Method: onChange
 Params:
 Return:
 Description:  This function is called on the onChange event. It checks
               for validations.
*****************************************************************************/


function _WidgetTextInput_OnChange(e)
{
   // no need to test for change on blur
   this.focusText = null;

   // Check the validation rule for the text.
   this.validateValue();

   // Resize the field if necessary.
   this.setSize(this.size);

   // Check if there is a callback set for this event.
   if (this.listenerObj != null)
   {
      // Check if the method is defined.
      if (this.listenerObj.onTextInputChange)
      {
         this.listenerObj.onTextInputChange(this);
      }
   }

}


/*****************************************************************************
 Class : WidgetTextInput
 Method: onBlur
 Params:
 Return:
 Description:  This function is called when the text box loses focus.
               It also checks for validations.
*****************************************************************************/


function _WidgetTextInput_OnBlur(e)
{
```

```
      // Test to see if value changed but no onChange call was made (IE bug)
      if ( this.focusText != null )
      {
         if ( this.getValue() != this.focusText )
705      {
            this.onChange( e );
         }
         this.focusText = null;
      }
710   this.applyOnBlurStyle();

      // Check if there is a callback set for this event.
      if (this.listenerObj != null)
      {
715      // Check if the method is defined.
         if (this.listenerObj.onTextInputBlur)
         {
            this.listenerObj.onTextInputBlur(this);
         }
720   }

      // This is needed because if the value of the element is changed during
      // the onChangeCallback event, the master value of the input box
      // is not reset.
725   this.element.value = this.element.value;

   }


730
   /*****************************************************************************
    Class : WidgetTextInput
    Method: onFocus
    Params:
735 Return:
    Description:  This function is called when the text box gets focus.
                  It also checks for validations.
   *****************************************************************************/

740

   function _WidgetTextInput_OnFocus(e)
   {
      var cjEvent = new CJEvent(e);
745
      if ((this.editable == true) && (this.disabled == false))
      {
         this.applyOnFocusStyle();

750      // pull out the curent text to save for blur test.
```

```
         // IE will not call onChange if you click outside the IFRAME!!
         this.focusText = this.getValue();

         // Check if there is a callback set for this event.
755      if (this.listenerObj != null)
         {
            // Check if the method is defined.
            if (this.listenerObj.onTextInputFocus)
            {
760            this.listenerObj.onTextInputFocus(this);
            }
         }
      }
      else
765   {
         if (this.disabled == true)
         {
            this.element.blur();
         }
770   }

   }


775
   /*****************************************************************************
    Class : WidgetTextInput
    Method: onKeyPress
    Params:
780 Return:
    Description:  This function is called when a key is pressed.
                  It also checks for validations.
   *****************************************************************************/

785

   function _WidgetTextInput_OnKeyPress(e)
   {
      var cjEvent = new CJEvent(e);
790
      return (this.validateKeyPress(cjEvent));
   }


795
   /*****************************************************************************
    Class : WidgetTextInput
    Method: validateKeyPress
    Params:
800 Return:
```

```
     Description:  Checks if the key pressed passes the validation rule. If it
                   fails the validation, the key press will not be allowed.
    ******************************************************************************/

805

    function _WidgetTextInput_ValidateKeyPress(cjEvent)
    {
       var keyChar;
810
       // If not the return key then process it.
       if (cjEvent.keyCode != 13)
       {
          keyChar = String.fromCharCode(cjEvent.keyCode);
815
          if (this.validationRule != null)
          {
             if (this.validationRule.isValidChar(keyChar) == false)
             {
820             return (false);
             }

             cjEvent.preventBubble();
          }
825    }

       return (true);
    }

830

    /******************************************************************************
     Class : WidgetTextInput
     Method: validateValue
835  Params:
     Return:
     Description:  Checks if the value currently in the text box passes the
                   validation rule.
    ******************************************************************************/
840

    function _WidgetTextInput_ValidateValue()
    {
845    this.valueIsValid = true;

       if (this.validationRule != null)
       {
          if (this.validationRule.isValidString(this.element.value) == false)
850       {
```

```
            this.valueIsValid = false;
        }
    }
855 CJSetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_INVALID, !this.valueIsValid);
}

860
/*****************************************************************************
 Event handlers
*****************************************************************************/

865

/*****************************************************************************
 Class : WidgetTextInput
 Method: InputElement_OnXXX
870 Params:
 Return: Child result.
 Description:  Static private methods use to invoke the object methods
               on the input element.
*****************************************************************************/
875

function _WidgetTextInput_InputElement_OnChange(e)
{
880 return (this.cjowner.onChange(e));
}

function _WidgetTextInput_InputElement_OnBlur(e)
{
885 return (this.cjowner.onBlur(e));
}

function _WidgetTextInput_InputElement_OnFocus(e)
{
890 return (this.cjowner.onFocus(e));
}

function _WidgetTextInput_InputElement_OnKeyPress(e)
{
895 return (this.cjowner.onKeyPress(e));
}


900 /*****************************************************************************
```

```
    Public Methods
    ****************************************************************************/


905
    WidgetTextInput.prototype.applyOnBlurStyle = _WidgetTextInput_ApplyOnBlurStyle;
    WidgetTextInput.prototype.applyOnFocusStyle = _WidgetTextInput_ApplyOnFocusStyle;
    WidgetTextInput.prototype.getDisabled = _WidgetTextInput_GetDisabled;
    WidgetTextInput.prototype.getEditable = _WidgetTextInput_GetEditable;
910 WidgetTextInput.prototype.getValue = _WidgetTextInput_GetValue;
    WidgetTextInput.prototype.invalidateValue = _WidgetTextInput_InvalidateValue;
    WidgetTextInput.prototype.isValueValid = _WidgetTextInput_IsValueValid;
    WidgetTextInput.prototype.setDisabled = _WidgetTextInput_SetDisabled;
    WidgetTextInput.prototype.setEditable = _WidgetTextInput_SetEditable;
915 WidgetTextInput.prototype.setListener = _WidgetTextInput_SetListener;
    WidgetTextInput.prototype.setSize = _WidgetTextInput_SetSize;
    WidgetTextInput.prototype.setStyleState = _WidgetTextInput_SetStyleState;
    WidgetTextInput.prototype.setValue = _WidgetTextInput_SetValue;
    WidgetTextInput.prototype.toElement = _WidgetTextInput_ToElement;
920 WidgetTextInput.prototype.focus = _WidgetTextInput_Focus;


    /****************************************************************************
925  Private Methods
    ****************************************************************************/


    WidgetTextInput.prototype.onChange = _WidgetTextInput_OnChange;
930 WidgetTextInput.prototype.onBlur = _WidgetTextInput_OnBlur;
    WidgetTextInput.prototype.onFocus = _WidgetTextInput_OnFocus;
    WidgetTextInput.prototype.onKeyPress = _WidgetTextInput_OnKeyPress;
    WidgetTextInput.prototype.validateValue = _WidgetTextInput_ValidateValue;
    WidgetTextInput.prototype.validateKeyPress = _WidgetTextInput_ValidateKeyPress;
935


    /****************************************************************************
     Public Methods definitions
940 ****************************************************************************/


    /****** END WidgetTextInput Class ***********************************************/
945


    /****************************************************************************
     Class : WidgetTextDisplay
950
```

```
    ******************************************************************************/

    /******************************************************************************
 955 Class : WidgetTextDisplay
     Method: WidgetTextDisplay   Constructor
     Params:
             String category                    -  This is a string identifying
             Anything value                     -  Initial value of the field.
 960 Return:
     Description:  Our text display widget.
    ******************************************************************************/


 965
    function WidgetTextDisplay(category, value)
    {
       // Create the container element.
       this.element = document.createElement("SPAN");
 970
       // Set some attributes.
       this.element.type = "text";

       // Create the text element.
 975   this.textElement = document.createTextNode(value);

       // Styles.
       // If category is not defined then use the default.
       if (!category)
 980   {
          category = WidgetTextDisplay.CATEGORY;
       }

       this.category = category;
 985
       CJSetClass(this.element, this.category + WidgetTextDisplay.STYLE_EXT_BASE, true);

       // Stick the text node into the span.
       this.element.appendChild(this.textElement);
 990 }


    /******************************************************************************
 995 Constants
    ******************************************************************************/


1000 // Category used if one is not specified by the user.
```

```
WidgetTextDisplay.CATEGORY = "WidgetTextDisplay";

/*****************************************************************************
   Basic style names.

   ".base"        = Base style for the control (Main container).
*****************************************************************************/


// Extensions for the styles.
WidgetTextDisplay.STYLE_EXT_BASE = "-base";


/*****************************************************************************
 Public Methods definitions
*****************************************************************************/


/*****************************************************************************
 Class : WidgetTextDisplay
 Method: getValue
 Params:
 Return: Any value;
 Description:  Returns this.element.value.
*****************************************************************************/


function _WidgetTextDisplay_GetValue()
{
   return (this.textElement.data);
}


/*****************************************************************************
 Class : WidgetTextDisplay
 Method: SetDisabled
 Params:
 Return: Any value;
 Description:
*****************************************************************************/


function _WidgetTextDisplay_SetDisabled()
```

```
    {
       // Does absolutely nothing right now.
    }

1055

    /*****************************************************************************
     Class : WidgetTextDisplay
     Method: setStyleState
1060 Params: String styleClass    -  Class of the style.
             Boolean state        -  state to set it to.
     Return:
     Description:  Sets the style state. The style must exist.
    *****************************************************************************/
1065


    function _WidgetTextDisplay_SetStyleState(styleClass, state)
    {
1070   CJSetClass(this.element, styleClass, state);
    }



1075 /*****************************************************************************
     Class : WidgetTextDisplay
     Method: setValue
     Params: Any value.
     Return: Any value;
1080 Description:  Sets the value of the field.
    *****************************************************************************/



1085 function _WidgetTextDisplay_SetValue(value)
     {
        if (value == null)
        {
           value = "";
1090    }

        this.textElement.data = value;
     }

1095

     /*****************************************************************************
      Class : WidgetTextDisplay
      Method: toElement
1100  Params:
```

```
     Return: HTMLElement
     Description:  Returns the main element associated with this widget.
    ************************************************************************/

1105

    function _WidgetTextDisplay_ToElement()
    {
       return (this.element);
1110 }


    /************************************************************************
1115  Private Methods definitions
    ************************************************************************/


1120 /************************************************************************
     Public Methods prototypes
    ************************************************************************/


1125
    WidgetTextDisplay.prototype.getValue = _WidgetTextDisplay_GetValue;
    WidgetTextDisplay.prototype.setDisabled = _WidgetTextDisplay_SetDisabled;
    WidgetTextDisplay.prototype.setStyleState = _WidgetTextDisplay_SetStyleState;
    WidgetTextDisplay.prototype.setValue = _WidgetTextDisplay_SetValue;
1130 WidgetTextDisplay.prototype.toElement = _WidgetTextDisplay_ToElement;


    /************************************************************************
1135  Private Methods prototypes
    ************************************************************************/


1140 /****** END WidgetTextDisplay Class ************************************/
```

Exhibit 10

```
/*****************************************************************************
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
       This document contains confidential/proprietary information.
*****************************************************************************

@(#)ControlTextInput.js
Initial Author: Jide Omisore

Description: Text Input Box control.

Dependencies:
  /util/browserapi.js
  /util/cjelementhelper.js
  /util/compat.js
  /controls/CJEvent.js
  /controls/exception.js
  /controls/iconhelper.js
  /controls/stylemgr.js
  /controls/widgettextinput.js


$Log: /green/webroot/common/controls/ControlTextInput.js $
 *
 * 29    11/29/99 3:14p Ddavidson
 * added focus() methods to focus to (first) text field
 *
 * 28    11/18/99 9:05p Jomisore
 * Fixed it so that 0 shows a plus sign when the force plus sign option is
 * set to true.
 *
 * 27    11/18/99 10:15a Tirish
 *
 * 26    10/05/99 10:14a Jomisore
 * Moved multiplyString to BrowserApi.js
 *
 * 25    9/24/99 10:48a Jomisore
 * The controls now take a category in the contstructor.
 *
 * 24    9/22/99 3:40p Jomisore
 * Made changes to the style stuff so that things run faster.
 *
 * 23    9/22/99 12:01p Sappling
 * This has been changed to partially implement the new styles
 *
 * 22    9/17/99 3:24p Wdover
```

```
     *
     * 21    9/10/99 4:47p Wdover
     * Modified to handle bad values
50   *
     * 20    9/09/99 9:49a Jomisore
     * Cleaned up code for the code review. Added the octet string control.
     * Added restrictions to the controls.
     *
55   * 19    9/01/99 3:57p Jomisore
     * Changed it so that the control's will display "?" and will be disabled
     * if primname is null.
     *
     * 18    8/23/99 2:36p Wdover
60   * Adding ControlBitString class
     *
     * 17    8/16/99 2:09p Jomisore
     * Changed getValueText() to toString() for the primitives.
     *
65   * 16    8/13/99 11:11a Jomisore
     * Fixed a problem with formatting when 0 is the value if the number edit
     * control.
     *
     * 15    8/11/99 9:20a Jomisore
70   * Added the "undefined" style for textdisplay widget.
     *
     * 13    8/09/99 5:41p Jomisore
     * Removed the set and get editable methods.
     *
75   * 12    8/09/99 9:17a Jomisore
     * Added a TextDisplayWidget.
     *
     * 11    8/06/99 11:43a Jomisore
     * Made the number formatting simplier.
80   *
     * 10    8/05/99 6:01p Jomisore
     * Changed it so that the digits specified that go to the left and right
     * of the decimal are formatting options and not restrictions. Although
     * when it is truncated the number is not rounded up (this will be worked
85   * on).
     *
     * 9     8/05/99 10:19a Jomisore
     * Updated a comment.
     *
90   * 8     8/04/99 5:49p Jomisore
     * Updated style management.
     *
```

```
     * 7     8/02/99 2:02p Jomisore
     * Finished up on the number edit box.
 95  *
     * 6     7/29/99 5:28p Jomisore
     * Made ControlTextInput a base class and added ControlCharStrEdit and
     * ControlNumberEdit to inherit from the base class.
     *
100  * 5     7/29/99 10:09a Jomisore
     * Changed "affectStyle" to "changeStyle".
     *
     * 4     7/28/99 10:46a Jomisore
     * Added the affectStyleById method.
105  *
     * 3     7/28/99 9:18a Jomisore
     * Added  some methods.
     *
     * 2     7/23/99 6:30p Jomisore
110  * Added comments.
     *
     * 1     7/23/99 6:26p Jomisore
     * Text Input Control.
     * 1     07/23/99 10:55p jomisore
115  * Initial implementation

    ******************************************************************************/


120
    /*******************************************************************************
     Class : ControlTextInput

             Public methods:
125             onTextInputChange
                setDisabled
                toElement
                valueChanged

130
    ******************************************************************************/

    /*******************************************************************************
     Class : ControlTextInput
135  Method: ControlTextInput  -  Constructor
     Params: Object prim            -  Primitive object.
             Boolean editable       -  Flags if this is editable
                                       or not.
```

```
            Boolean disabled        -  Flags if this is disabled
140                                    or not.
            Number tabIndex         -  Tab index.
            String category         -  Category used by the text input widget
                                       if one is created.
            String textSpanCategory -  Category used by the text display widget
145                                    if one is created.
            Number size             -  Size of the text field in characters.
            Number maxLength        -  Maximum number of characters that the user
                                       can input.
            ValidationManager validationMgr -   Validation manager.
150         Boolean validateOnKeyPress -  Validate each key stroke?
    Return:
    Description: Base class for the text input controls. The following controls
                 should inherit from it:

155              ControlCharStrEdit
                 ControlNumberEdit
                 ControlBitStrEdit
                 ControlOctetStrEdit

160 ****************************************************************************/
    function ControlTextInput(prim, editable, disabled, tabIndex, category,
                              size, maxLength, validationMgr,
                              validateOnKeyPress)
    {
165    var message;
       var undefined;

       this.ControlBase(editable, disabled, tabIndex);
       // Required fields.
170    // Find the primitive.
       this.prim = prim;

       if (!this.prim)
       {
175       // If this parameter is undefined then this is being called when the
          // prototype object for the derived controls are being initialized.
          if (prim === undefined)
          {
             return;
180       }
          else
          {
             size = 0;
             this.disabled = true;
```

```
185         }
        }
        else
        {
           // Add this control to the primitive's list of listeners.
190        this.prim.addListener(this);

           if (this.prim.isWritable() == false)
           {
              // If the primitive is not writable then force the editable to be false.
195           this.editable = false;
           }

        }

200     this.category = category;

        if ((editable == true) || (editable == null))
        {
           this.textWidgetCategory = "-" + WidgetTextInput.CATEGORY;
205
           // Add the text widget.
           this.widgets[0] = new WidgetTextInput(this.editable, this.disabled, this.tabIndex,
                                                 validationMgr, validateOnKeyPress,
                                                 this.category + this.textWidgetCategory,
210                                              size, maxLength);
           this.oldTabIndex = this.tabIndex;
           this.widgets[0].setListener(this);
        }
        else
215     {
           this.textWidgetCategory = "-" + WidgetTextDisplay.CATEGORY;

           // Use a text display instead of a text input.
           this.widgets[0] = new WidgetTextDisplay(this.category + this.textWidgetCategory);
220     }

        // Fill in the value of the primitive if there is one.
        this.valueChanged();
    }
225


    /*****************************************************************************
     Constants
230 *****************************************************************************/
```

```
    /*****************************************************************************
       Basic style extension.
235    "-writable"    = Writable style for the control's widget.
       "-invalid"     = Invalid style for the control's widget.
       "-focus"       = Focus style for the control's widget.
       "-disabled"    = Disabled style for the control's widget.
       "-undefined"   = Undefined style for the control's widget.
240    "-badvalue"    = Badvalue style for the control's widget.
    *****************************************************************************/


245 // Extensions for the styles.
    ControlTextInput.STYLE_EXT_BASE = "-base";
    ControlTextInput.STYLE_EXT_WRITABLE = "-writable";
    ControlTextInput.STYLE_EXT_INVALID = "-invalid";
    ControlTextInput.STYLE_EXT_FOCUS = "-focus";
250 ControlTextInput.STYLE_EXT_DISABLED = "-disabled";
    ControlTextInput.STYLE_EXT_UNDEFINED = "-undefined";
    ControlTextInput.STYLE_EXT_BADVALUE = "-badvalue";


255
    /*****************************************************************************
     Public Methods definitions
    *****************************************************************************/

260

    /*****************************************************************************
     Class : ControlTextInput
     Method: onTextInputChange
265  Params:
     Return:
     Description:  This is called by the text widget when the value of the text
                   input box changes.
    *****************************************************************************/
270


    function _ControlTextInput_OnTextInputChange()
    {
275    var value;
```

```
      CJAssert(this.prim, "There has to be a primitive defined.");

      if (this.widgets[0].isValueValid())
280   {
         value = this.widgets[0].getValue();
         this.valid = true;
         try
         {
285         this.prim.setValue(value, this);

            // Clear the undefined state.
            if (value != null)
            {
290            this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                             + ControlTextInput.STYLE_EXT_UNDEFINED,
                                             false);
            }
         }
295      catch (ex)
         {
            if (ex instanceof CJException.InvalidValue)
            {
               this.widgets[0].invalidateValue();
300            this.valid = false;
            }
            else
            {
               throw (ex);
305         }
         }
      }
      else
      {
310      this.valid = false;
      }
   }


315
   /*****************************************************************************
    Class : ControlTextInput
    Method: setDisabled
    Params:
320 Return:
    Description:  Gets the disabled attribute.
   *****************************************************************************/
```

```
325
    function _ControlTextInput_SetDisabled(disabled)
    {
       CJAssert((disabled == false) || (disabled == true),
                "disabled must be true or false");
330
       this.disabled = disabled;

       this.widgets[0].setDisabled(this.disabled);

335 }


    /*****************************************************************************
340  Class : ControlTextInput
     Method: toElement
     Params:
     Return:
     Description:  Returns the main element.
345 *****************************************************************************/


    function _ControlTextInput_ToElement()
350 {
       return (this.widgets[0].toElement());
    }


355 /*****************************************************************************
     Class : ControlTextInput
     Method: focus
     Params:
     Return:
360  Description:  Give the control the focus
    *****************************************************************************/


365 function _ControlTextInput_Focus()
    {
       return (this.widgets[0].focus());
    }
```

```
370
    /*****************************************************************************
     Class : ControlTextInput
     Method: valueChanged
375  Params:
     Return:
     Description:  This is a callback that is registered with the associated
                   primitive that is called when the value of the primitive changes.
    *****************************************************************************/
380

    function _ControlTextInput_ValueChanged()
    {
385    var value;

       if (this.prim)
       {
          if (this.prim.hasBadValue())
390       {
             this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                           + ControlTextInput.STYLE_EXT_BADVALUE, false);

             this.badValue = true;
395       }
          else if (this.badValue)
          {
             this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                           + ControlTextInput.STYLE_EXT_BADVALUE, false);
400
             this.badValue = false;
          }
          // If the value is undefined then the value returned will be "".
          value = this.getPrimValueText(this.prim);
405
          if (this.checkUndefined() == false)
          {
             this.widgets[0].setValue(value);
          }
410       else
          {
             this.widgets[0].setValue("?");
          }
```

```
415     }
        else
        {
           this.widgets[0].setValue("?");
        }
420
    }


425 /*****************************************************************************
     Private Methods definitions
    *****************************************************************************/


430
    /*****************************************************************************
     Class : ControlTextInput
     Method: checkUndefined
     Params:
435  Return: Boolean true or false.
     Description:  Checks if the primitive is undefined. If it is, it sets the
                   undefined adapter state to true.
    *****************************************************************************/

440

    function _ControlTextInput_CheckUndefined()
    {
       var undefined;
445
       CJAssert(this.prim, "There has to be a primitive defined.");

       undefined = this.prim.isUndefined();

450    this.widgets[0].setStyleState(this.category + this.textWidgetCategory +
                                     ControlTextInput.STYLE_EXT_UNDEFINED, undefined);

       return (undefined);
    }
455

    /*****************************************************************************
     Class : ControlTextInput
     Method: getPrimValueText
460  Params:
```

```
     Return: primitve value.
     Description:  Get's the text value of the prim. Override this to so any special
                   formatting with the primitive value.
    ************************************************************************************/
465


    function _ControlTextInput_getPrimValueText(prim)
    {
470    var value;

       CJAssert(prim, "There has to be a primitive defined.");

       value = prim.toString();
475
       return (value);
    }


480
    /************************************************************************************
     Public Methods prototypes
    ************************************************************************************/

485


    ControlTextInput.prototype = new ControlBase();
    ControlTextInput.prototype.ControlTextInput = ControlTextInput;

490 ControlTextInput.prototype.onTextInputChange = _ControlTextInput_OnTextInputChange;
    ControlTextInput.prototype.setDisabled = _ControlTextInput_SetDisabled;
    ControlTextInput.prototype.toElement = _ControlTextInput_ToElement;
    ControlTextInput.prototype.focus = _ControlTextInput_Focus;
    ControlTextInput.prototype.valueChanged = _ControlTextInput_ValueChanged;
495


    /************************************************************************************
     Private Methods prototypes
500 ************************************************************************************/



    ControlTextInput.prototype.checkUndefined = _ControlTextInput_CheckUndefined;
505 ControlTextInput.prototype.getPrimValueText = _ControlTextInput_getPrimValueText;
```

```
/****** END ControlTextInput Class ******************************************/


/*****************************************************************************
 Class : ControlCharStrEdit
 Method: ControlCharStrEdit - Constructor
 Params: String primname                    -  Name of the primitive.
         Boolean editable                   -  Flags if this is editable
                                               or not.
         Boolean disabled                   -  Flags if this is disabled
                                               or not.
         Number tabIndex                    -  Tab index.
         Number size                        -  Size of the text field.
 Return:
 Description:  Character String edit Control.
*****************************************************************************/


function ControlCharStrEdit(primname, editable, disabled, tabIndex, size, category)
{
   var prim;
   var maxLength;

   if (category == null)
   {
      category = ControlCharStrEdit.CATEGORY;
   }

   if (primname == null)
   {
      prim = null;
   }
   else
   {
      // Get the prim.
      prim = PrimitiveBase.findPrimitive(primname);

      // Make sure we have the correct type for the primitive.
      if (prim instanceof PrimitiveString)
      {
         // This is ok.
      }
```

```
         else
         {
555         // Bad primitive.
            message = "Primitive \"" + primname + "\" type is invalid for Number Edit controls";

            throw new CJException.InvalidValue(message);
         }
560
         // Get the maximum number of characters that can be entered.
         if (prim.hasMaxLength())
         {
            maxLength = prim.getMaxLength();
565      }
      }

      // Initialize the base class.
      this.ControlTextInput(prim, editable, disabled, tabIndex,
570                           category, size, maxLength);
   }



575 // Inherit from text input control.
   ControlCharStrEdit.prototype = new ControlTextInput();


580 /*****************************************************************************
    Constants
   *****************************************************************************/


585
   // Styles.
   ControlCharStrEdit.CATEGORY = "ControlCharStr";


590
   /****** END ControlCharStrEdit Class ***************************************/



595 /*****************************************************************************
    Class : ControlNumberEdit
    Method: ControlNumberEdit - Constructor
    Params: String primname                    -  Name of the primitive.
```

```
            Boolean editable                    -  Flags if this is editable
600                                                or not.
            Boolean disabled                    -  Flags if this is disabled
                                                   or not.
            Number tabIndex                     -  Tab index.
            Number size                         -  Size of the text field in characters.
605         Number digLeftOfDecimal             -  Minimum number of digits to the
                                                -  left of the decimal point.
            Number digRightOfDecimal            -  The number of digits that have
                                                -  to be to the right of the
                                                -  decimal point.
610         Boolean showPlusSign                -  Boolean indicating whether to
                                                   automatically show a plus sign
                                                   or not.
     Return:
     Description:  Number edit Control.
615 ***************************************************************************/


    function ControlNumberEdit(primname, editable, disabled, tabIndex, size,
620                            digLeftOfDecimal, digRightOfDecimal, showPlusSign, category)
    {

       var prim;
       var validationRule = null;
625    var validationMgr = null;
       var primitiveMax;
       var primitiveMin;
       var primitivePrecision = null;

630    if (category == null)
       {
          category = ControlNumberEdit.CATEGORY;
       }

635    if (primname == null)
       {
          prim = null;
          digLeftOfDecimal = 0;
          digRightOfDecimal = null;
640       showPlusSign = false;
       }
       else
       {
          // Get the prim.
```

```
645         prim = PrimitiveBase.findPrimitive(primname);

            // Make sure we have the correct type for the primitive.
            if (prim instanceof PrimitiveNumber)
            {
650            // This is ok.
            }
            else
            {
               // Bad primitive.
655            message = "Primitive \"" + primname + "\" type is invalid for Number Edit controls";

               throw new CJException.InvalidValue(message);
            }

660         // Get restrictions.
            primitiveMin = prim.getMinimum();
            primitiveMax = prim.getMaximum();

            if (prim.hasPrecision() == true)
665         {
               primitivePrecision = prim.getPrecision();
            }

            validationRule = new RuleRange(primitiveMin, primitiveMax);
670         validationMgr = new ValidationManager([validationRule]);

         }

         // Formatting options.
675      if (!digLeftOfDecimal)
         {
            digLeftOfDecimal = 0;
         }

680      this.digLeftOfDecimal = digLeftOfDecimal;
         this.digRightOfDecimal = digRightOfDecimal;

         if (primitivePrecision != null)
         {
685         this.digRightOfDecimal = primitivePrecision;
         }

         this.showPlusSign = false;

690      if (showPlusSign)
```

```
    {
       this.showPlusSign = showPlusSign;
    }

695 // Initialize the base class.
    this.ControlTextInput(prim, editable, disabled, tabIndex,
                          category,  size, null,
                          validationMgr,
                          true);
700 }


    // Inherit from text input control.
705 ControlNumberEdit.prototype = new ControlTextInput();


    /*****************************************************************************
710  Constants
    *****************************************************************************/


715 // Styles.
    ControlNumberEdit.CATEGORY = "ControlNumberEdit";


720 /*****************************************************************************
     Public Methods definitions
    *****************************************************************************/


725
    /*****************************************************************************
     Class : ControlNumberEdit
     Method: getPrimValueText
     Params:
730  Return:
     Description:  Get's the text value of the prim. It also formats the value
                   according to what th user wants.
    *****************************************************************************/

735
```

```
function _ControlNumberEdit_getPrimValueText(prim)
{
   var value;

   CJAssert(prim, "There has to be a primitive defined.");

   value = prim.toString();

   value = this.formatValue(value);

   return (value);
}


/*****************************************************************************
 Class : ControlNumberEdit
 Method: onTextInputChange
 Params:
 Return:
 Description:  This is called by the text widget when the value of the text
               input box changes.
*****************************************************************************/


function _ControlNumberEdit_OnTextInputChange()
{
   var stringValue;

   CJAssert(this.prim, "There has to be a primitive defined.");

   stringValue = this.widgets[0].getValue();

   // If the text entered is invalid then do not bother to set the primitive.
   if (this.widgets[0].isValueValid() == true)
   {
      stringValue = this.formatValue(stringValue);

      // Set the value of the text widget.
      this.widgets[0].setValue(stringValue);
      this.valid = true;
      try
      {
         // Set the value of the primitive.
         this.prim.setValue(stringValue, this);
```

```
            // Clear the undefined state.
785         if (stringValue != null)
            {
               this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                             + ControlTextInput.STYLE_EXT_UNDEFINED, false);
            }
790      }
         catch (ex)
         {
            if (ex instanceof CJException.InvalidValue)
            {
795            this.widgets[0].invalidateValue();
               this.valid = false;
            }
            else
            {
800            throw (ex);
            }
         }

      }
805   else
      {
         this.valid = false;
      }

810 }


   /*****************************************************************************
815  Private Methods definitions
   *****************************************************************************/



820 /*****************************************************************************
     Class : ControlNumberEdit
     Method: formatValue
     Params: String invalue    -  String to format.
     Return:
825  Description:  Formats the value in the text field to however the user
                   specified it in the constructor. The inut value has to be a
                   valid javascript number.
   *****************************************************************************/
```

```
830
    function _ControlNumberEdit_FormatValue(invalue)
    {
       var value = invalue;
835    var numberLiteral;

       if ((value != null) && (value != ""))
       {
          // Put the decimal point in the right spot.
840       value = cjCompat.floatFix(value, this.digLeftOfDecimal, this.digRightOfDecimal);

          // Add any plus signs.
          // If the user wants to stick in the plus sign in front of positive
          // numbers then do it.
845       if (this.showPlusSign == true)
          {
             numberLiteral = new Number(value);

             if (numberLiteral >= 0.0)
850          {
                // If plus sign isn't already there then stick it in.
                if (value.indexOf("+", 0) == -1)
                {
                   value = "+" + value;
855             }
             }
          }

          return (value.toString());
860    }
       else
       {
          return (value);
       }
865 }



870

    /**************************************************************************
     Public Methods prototypes
```

```
875 ***************************************************************************/


   ControlNumberEdit.prototype.getPrimValueText = _ControlNumberEdit_getPrimValueText;
880 ControlNumberEdit.prototype.onTextInputChange = _ControlNumberEdit_OnTextInputChange;


   /***************************************************************************
885  Private Methods prototypes
   ***************************************************************************/


890 ControlNumberEdit.prototype.formatValue = _ControlNumberEdit_FormatValue;


   /****** END ControlNumberEdit Class ****************************************/
895
   /*************************************************************************
    Class : ControlBitStrEdit

    This class provides a control for editing bit strings of variable
900 lengths.

    PUBLIC METHODS:
      onTextInputChange

905 *************************************************************************/

   /***************************************************************************
    Class : ControlBitStrEdit
    Method: ControlBitStrEdit  -  Constructor
910 Params: String primname          -  Name of bit string primitive.
            Boolean editable         -  Flags if this is editable
                                        or not.
            Boolean disabled         -  Flags if this is disabled
                                        or not.
915         Number tabIndex          -  Tab index.
            Number size              -  Size of the text field in characters.
    Return:

   ***************************************************************************/
920
```

```
function ControlBitStrEdit(primname, editable, disabled, tabIndex, size, category)
{
   var prim;
   var maxLength;
   var validationRule = null;
   var validationMgr = null;

   if (category == null)
   {
      category = ControlBitStrEdit.CATEGORY;
   }

   if (primname == null)
   {
      prim = null;
   }
   else
   {
      prim = PrimitiveBase.findPrimitive(primname);

      if ((prim instanceof PrimitiveBitString) == false)
      {
         message = "Primitive \""+primname+"\" type is invalid for Bit String Edit Controls.";
         throw new CJException.InvalidValue(message);
      }

      validationRule = new RuleBase(/^[\?]|[01]*$/, /^[01]$/);
      validationMgr = new ValidationManager([validationRule]);

      // Get the maximum number of characters that can be entered.
      maxLength = prim.getMaxBits();

   }

   this.ControlTextInput(prim, editable, disabled, tabIndex,
                         category, size, maxLength,
                         validationMgr,true);

}

// Inherit from text input control
ControlBitStrEdit.prototype = new ControlTextInput();


/*****************************************************************************
```

```
     Constants
    ************************************************************************/

970 // Styles.
    ControlBitStrEdit.CATEGORY = "ControlBitStr";


975 /****** END ControlBitStrEdit Class ****************************************/


    /**************************************************************************
980  Class : ControlOctetStrEdit

     This class provides a control for editing octet strings of variable
     lengths.

985  PUBLIC METHODS:
       onTextInputChange

    **************************************************************************/

990 /**************************************************************************
     Class : ControlOctetStrEdit
     Method: ControlOctetStrEdit  -  Constructor
     Params: String primname          -  Name of octet string primitive.
             Boolean editable         -  Flags if this is editable
995                                      or not.
             Boolean disabled         -  Flags if this is disabled
                                         or not.
             Number tabIndex          -  Tab index.
             Number size              -  Size of the text field in characters.
1000         Boolean separateBytes    -  Whether to separate the bytes with spaces.
                                         Default is false.
     Return:

    **************************************************************************/
1005
    function ControlOctetStrEdit(primname, editable, disabled, tabIndex, size,
                                 separateBytes, category)
    {
       var prim;
1010   var maxLength;
       var primMaxLength;
       var validationRule = null;
```

```
        var validationMgr = null;

1015    if (category == null)
        {
           category = ControlOctetStrEdit.CATEGORY;
        }

1020    this.separateBytes = separateBytes;

        if (primname == null)
        {
           prim = null;
1025    }
        else
        {
           prim = PrimitiveBase.findPrimitive(primname);

1030       if ((prim instanceof PrimitiveOctetString) == false)
           {
              message = "Primitive \""+primname+"\" type is invalid for Octet String Edit Controls.";
              throw new CJException.InvalidValue(message);
           }
1035
           validationRule = new RuleOctet(2, true, true);
           validationMgr = new ValidationManager([validationRule]);

           // Get the maximum number of characters that can be entered.
1040       // Max length will be maximum number of bytes * 2.
           primMaxLength = prim.getMaxLength();
           maxLength = primMaxLength * 2;

           // Factor in the spaces for separating the bytes..
1045       if (this.separateBytes == true)
           {
              maxLength += primMaxLength - 1;
           }

1050    }

        this.ControlTextInput(prim, editable, disabled, tabIndex,
                              category, size, maxLength,
                              validationMgr,true);
1055
    }

    // Inherit from text input control
```

```
ControlOctetStrEdit.prototype = new ControlTextInput();

/*****************************************************************************
 Constants
*****************************************************************************/

// Styles.
ControlOctetStrEdit.CATEGORY = "ControlOctetStr";


/*****************************************************************************
 Public Methods definitions
*****************************************************************************/


/*****************************************************************************
 Class : ControlOctetStrEdit
 Method: getPrimValueText
 Params:
 Return:
 Description:  Get's the text value of the prim. It also formats the value
               according to what th user wants.
*****************************************************************************/


function _ControlOctetStrEdit_getPrimValueText(prim)
{
   var value;

   CJAssert(prim, "There has to be a primitive defined.");

   value = prim.toString();

   value = this.formatValue(value);

   return (value);
}


/*****************************************************************************
```

```
1105  Class : ControlOctetStrEdit
      Method: onTextInputChange
      Params:
      Return:
      Description:  This is called by the text widget when the value of the text
1110                input box changes.
     ********************************************************************************/


1115 function _ControlOctetStrEdit_OnTextInputChange()
     {
        var stringValue;

        // Get the value entered.
1120    stringValue = this.widgets[0].getValue();

        // Add or remove spaces.
        stringValue = this.formatValue(stringValue);

1125    // Set the value of the text widget with the formated string.
        this.widgets[0].setValue(stringValue);

        // If the text entered is invalid then do not bother to set the primitive.
        if (this.widgets[0].isValueValid() == true)
1130    {
           // Remove spaces and stick it into the primitive.
           stringValue = stringValue.replace(new RegExp("[ ]", "g"), "");
           this.valid = true;
           try
1135       {
              // Set the value of the primitive.
              this.prim.setValue(stringValue, this);

              // Clear the undefined state.
1140          if (stringValue != null)
              {
                 this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                               + ControlTextInput.STYLE_EXT_UNDEFINED, false);
              }
1145       }
           catch (ex)
           {
              if (ex instanceof CJException.InvalidValue)
              {
1150             this.widgets[0].invalidateValue();
```

```
                this.valid = false;
            }
            else
            {
1155            throw (ex);
            }
        }

    }
1160 else
    {
        this.valid = false;
    }

1165 }


    /*****************************************************************************
     Private Methods definitions
    *****************************************************************************/


1175 /*****************************************************************************
     Class : ControlOctetStrEdit
     Method: formatValue
     Params: String invalue    -  String to format.
     Return:
1180 Description:  Formats the value in the text field to however the user
                   specified it in the constructor. The inut value has to be a
                   valid hex string.
    *****************************************************************************/

1185

    function _ControlOctetStrEdit_FormatValue(stringValue)
    {
        var newValue = "";
1190    var chr;
        var nibbleCount;

        CJAssert(this.prim, "There has to be a primitive defined.");

1195
        // Number of nibbles found.
```

```
        nibbleCount = 0;

        // Format now.
1200    for (var index = 0; index < stringValue.length; index++)
        {
           chr = stringValue.charAt(index);

           if (chr != " ")
1205       {
              newValue += chr;

              ++nibbleCount;

1210          // This is a byte.
              if (nibbleCount == 2)
              {
                 // If user wants to separate the bytes then add a space.
                 if (this.separateBytes == true)
1215             {
                    newValue += " ";
                 }

                 nibbleCount = 0;
1220          }
           }
        }

        return (newValue);
1225 }


     /*****************************************************************************
1230  Public Methods prototypes
     *****************************************************************************/


1235 ControlOctetStrEdit.prototype.getPrimValueText = _ControlOctetStrEdit_getPrimValueText;
     ControlOctetStrEdit.prototype.onTextInputChange = _ControlOctetStrEdit_OnTextInputChange;


1240 /*****************************************************************************
      Private Methods prototypes
     *****************************************************************************/
```

```
1245
     ControlOctetStrEdit.prototype.formatValue = _ControlOctetStrEdit_FormatValue;


1250 /****** END ControlOctetStrEdit Class *****************************************/
```

Exhibit 11

```
/*============================================================================
                        AUTOMATED LOGIC CORPORATION
                Copyright (c) 1999 - 2000 All Rights Reserved
        This document contains confidential/proprietary information.
============================================================================

    @(#)PrimitiveUpdateServlet.java

    Author(s) ddavidson

    $Log: /green/source/com/controlj/green/web/servlets/PrimitiveUpdateServlet.java $
 *
 * 10    12/09/99 1:50p Jomisore
 * Made getChangesFromCore a method in ServletBase.
 *
 * 9     11/24/99 1:32p Jomisore
 * Added more reserved parameters for callbacks.
 *
 * 8     11/22/99 2:18p Tirish
 *
 * 7     10/28/99 4:16p Ddavidson
 * added 'false' argument to req.getSesssion( false ) to avoid created new
 * session when not wanted.
 *
 * 6     10/11/99 11:53a Wdover
 *
 * 5     9/28/99 2:52p Wdover
 *
 * 4     9/03/99 2:12p Wdover
 * Modified to gracefully exit after HTTPSession timeout
 *
 * 3     9/03/99 11:28a Sappling
 * Added JavaScript call to trigger reload
 *
 * 2     8/26/99 10:46a Ddavidson
 * Added Submit/Update servlet for primitives.
 * Moved stdHeader/Footer/Content methods into ServletBase

============================================================================*/

package com.controlj.green.web.servlets;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.controlj.green.web.servlets.*;
import com.controlj.green.web.servlets.util.*;
import com.controlj.green.web.coreaccess.PrimitiveBinder;

```

```
/**<!=============================================================================>
   PrimitiveUpdateServlet is resposible for retrieving action page changes
   from a PrimitiveBinder and generating the JavaScript code that will
   notify that client of the new values via JscriptSetMasterValue objects.
<!=============================================================================>*/

public class PrimitiveUpdateServlet extends ServletBase
{
   /**<!===== doGet ============================================================>
      Handle HTTP GET requests.  Serves resoruce atfer verifying session.
      The resources is any file specified by the "path" following the
      servlets URL.  For example "/servlet/protector/dir/x.htm" will serve
      the "dir/x.htm" file in the configured PRO_PROTDIR location.

      @param req  The standard HttpServletRequest
      @param resp The standard HttpServletResponse
      @author  ddavidson
   <!=============================================================================>*/

   public void doGet(HttpServletRequest req, HttpServletResponse resp)
   throws ServletException, IOException
   {
      String afterUpdateCallback = null;

      debugInfo( "PrimitiveUpdate:doGet - HTML" );
      ////////////////////////////////////////////////////////////////////////////
      ////////////////////////// Extract parameters //////////////////////////////
      ////////////////////////////////////////////////////////////////////////////
      String getValuesFromField = req.getParameter( "getFieldValues")  ;

      WebBrowserSession wbs = null;
      try
      {
         wbs = verifySession( req, true );
      }
      catch (ServletException e)
      {
         if (req.getSession(false) == null)
         {
            redirectTimeout(resp);
            return;
         }
         else
            throw e;
      }

      afterUpdateCallback = req.getParameter("cjAfterUpdateCallback");

      PrimitiveBinder pb = (PrimitiveBinder)wbs.get( "PrimitiveBinder" );
```

```
            ////////////////////////////////////////////////////////////////////////
            ////////////////////// setup response list //////////////////////////////
            ////////////////////////////////////////////////////////////////////////

105         JscriptMsgList list = new JscriptMsgList( );

            // we are responsible for PrimitiveBinder synchronicity
            if ( pb != null )
            {
110            getChangesFromCore(pb, getValuesFromField, list);
            }

            list.addItem( new JscriptSimple("top.setActBusy(false);"));

115         if (afterUpdateCallback != null)
            {
               list.addItem( new JscriptSimple(afterUpdateCallback + "();"));
            }
            else
120         {
               list.addItem( new JscriptSimple("window.parent.updateFinished();"));
            }

            ////////////////////////////////////////////////////////////////////////
125         ////////////// Now generate output stream and setup content ////////////
            ////////////////////////////////////////////////////////////////////////

            stdContent( req, resp, null, list, null );
         }
130
         /**<!===== doPost ============================================================>
            POST redirects to "GET"
            @param  resp The HttpServletResponse
            @author  ddavidson
135      <!============================================================================>*/

         public void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException
         {
140         doGet( req, resp );
         }
         ////////////////////////////////////////////////////////////////////////////
         ////////////////////////////// Standard debug //////////////////////////////
         ////////////////////////////////////////////////////////////////////////////
145
         private void debugInfo(String message)  { com.controlj.green.common.Debug.info(
"PrimitiveUpdateServlet",message); };
         private void debugError(String message) {
com.controlj.green.common.Debug.error("PrimitiveUpdateServlet",message); };
      }
```

Exhibit 12

```
/*============================================================================
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
      This document contains confidential/proprietary information.
============================================================================

    @(#)ServletBase.java

    Author(s) ddavidson

    $Log: /green/source/com/controlj/green/web/servlets/ServletBase.java $
 *
 * 16    12/09/99 1:48p Jomisore
 * Added getChangesFromCore so that the submit and update servlets can
 * both call it.
 *
 * 15    10/28/99 4:15p Ddavidson
 * fixed expireBrowserSession to handle 'null' login context
 *
 * 14    10/11/99 11:53a Wdover
 *
 * 13    9/28/99 2:52p Wdover
 *
 * 12    9/17/99 4:51p Wdover
 * Fixed problem with CJException
 *
 * 11    9/13/99 8:53a Ddavidson
 * changed login exception handling.  removed relative path name in
 * redirectTimeout.
 *
 * 10    9/10/99 10:22a Ddavidson
 * added getWebLoginContext() method
 *
 * 9     9/09/99 11:30a Ddavidson
 * First pass at adding security via Operator and PrivilegeSet.
 * Nothing is checking privileges yet.
 *
 * 8     9/03/99 2:12p Wdover
 * Adding redirectTimeout method
 *
 * 7     8/26/99 5:02p Sappling
 *
 * 6     8/26/99 10:46a Ddavidson
 * Added Submit/Update servlet for primitives.
 * Moved stdHeader/Footer/Content methods into ServletBase
 *
```

```
    * 5     8/24/99 9:41a Ddavidson
    * Added "addStdJavaScriptReturn" method.
    *
50  * 4     8/20/99 4:35p Ddavidson
    * Removed local debug viewer
    *
    * 3     8/12/99 10:15a Ddavidson
    * Rewrite using new core for Milestone 2
55  *
    * 2     8/09/99 2:46p Ddavidson
    * web browser session
    *
    * 1     6/30/99 9:27a Ddavidson
60
   ============================================================================*/

   package com.controlj.green.web.servlets;

65 import java.io.*;
   import java.util.*;
   import javax.servlet.*;
   import javax.servlet.http.*;
   import com.controlj.green.common.*;
70 import com.controlj.green.web.coreaccess.*;
   import com.controlj.green.web.servlets.util.*;
   import com.controlj.green.core.data.*;

   /**<!============================================================================>
75   Base class under which all SVWeb specific servlets should derive
     themselves.  Contains helper methods that are available to all servlet
     subclasses.  Before serving up a session specific data, the servlet
     should call the verifySession() method to verify that the current
     session, if any, is valid.  If so, a WebBrowserSession will be returned
80   that may be used to manage any user-session specific information.
     <P>
     For each HttpSession, the is exactly one WebLoginContext object.
     There may be many WebBrowserSession objects as well.  There should be exactly
     one for each browser window that is open at a time.  Each servlet
85   request must identify itself with a "wbs=#" parameter as part of each
     request.  If a browser window is "duplicated", a new user context ID
     will be assigned to the new window as part of the JavaScript response.
     <P>
     VerifyContext will automatically handle the case where a user performs
90   a "browser: open new window".  A second window requires a new user
     context.
     <P>
```

```
    To catch a browser "refresh" operation, the servlet must coordinate a
    sequence number mechanism with the companion javascript. For each request
 95 the javascript makes (by loading a new URL into a sub frame) the
    sequence number should increment.  If the sequence number is that same
    as the previous request, the servlet should assume that the user has
    selected "refresh" and act as thought the master page was reloaded which
    has the effect of destroying the previous javascript environment.
100 <P>
    For any items that are cached, the servlet must maintain an "ackKey"
    in the user context and associate it with any elements sent to javascript.
    <B>This requires tight coupling with the associated javascript code</B>
    <P>
105 Typically, each servlet that cares about WebBrowserSession and sending only
    "changed data" to the javascript will do most of the following:<br>
    <br>
    <CODE>
      WebBrowserSession wbs = verifySession( req, getServletConfig(), true );<br>
110   if ( wbs.isNew() )<br>
      {<br>
           initSequence( wbs, "myAckKey" );<br>
           // do other stuff - user has done "browser refresh"<br>
      }<br>
115   if ( isRepeatedSequence( wbs, "mySeqKey", req.getParameter( "seq" )))<br>
      {<br>
           initSequence( wbs, "myAckKey" );<br>
           // user has done "new window"<br>
      }<br>
120   ...<br>
      // acknowlege receipt of previous cached items sent<br>
      ackCache( wbs, req.getParameter( "ack" ) );<br>
      ...<br>
      String nextAck = incrementSequence( wbs, "myAckKey" );<br>
125   JscriptMsgList list = new JscriptMsgList( nextAck );<br>
      ...// build up response<br>
      list.write( out ); // write the javascript <br>

    </CODE>
130  @author ddavidson.
  <!=============================================================================>*/

  public class ServletBase extends HttpServlet
  {
135   /**<!===== login ============================================================>
        Attempt to login to SVWeb and setup the user context information.
        This will perform a login to the core database as well.
```

```
      @param  req      The request triggering the login
140   @param  sc       The login servlet's configuration.  Provides session global params.
      @param  user     The user name to verify.
      @param  pass     The supplied user password.
      @return a WebLoginContext if login was successful, else 'null'.
      @author  ddavidson
145   <!=============================================================================>*/

      protected WebLoginContext login( HttpServletRequest req, ServletConfig sc,
                                       String user, String pass )
            throws com.controlj.green.common.CJException
150   {
         // be sure to clean up any old info
         // invalidateSession( req );

         Operator operator = Operator.getOperator( user, pass );
155      user = operator.getLoginName();

         // create a new session object and a WebLoginContext
         HttpSession session = req.getSession( true );
         return createWebLoginContext( req, sc, session, operator, user );
160   }

      /**<!===== verifySession ================================================>
         Verify that we have a valid sesion and return the WebBrowserSession.

165      Verifies that an HttpSession exists and that it contains a
         WebBrowserSession.  Throws an exception if either is invalid.  The request
         must contain a parameter identifying the WebBrowserSession via "wbs".

         @param  req     The HttpServletRequest to be verified
170      @param  create  True if we should create a new WebBrowserSession if one not found.
         @return a WebBrowserSession
         @author  ddavidson
      <!=============================================================================>*/

175   static protected WebBrowserSession verifySession( HttpServletRequest req,
            boolean create )
         throws ServletException
      {
         return WbsBean.verifySession( req, create );
180   }

      /**<!===== invalidateSession ============================================>
         Invalidates the current HttpSession.
```

```
185       Further access using the current session ID will fail.  This should
          not normally be called by a servlet.

          @param  req
          @author  ddavidson
190       <!=========================================================================>*/

          protected void invalidateSession( HttpServletRequest req )
          {
             HttpSession session = req.getSession( false );
195          if ( session != null )
                session.invalidate();
          }

          /**<!===== expireWebBrowserSessions ====================================>
200          Scan the WebBrowserSession for this HttpSession and destroy any that are
             older than out timeout property.
             @param  req       HttpRequest triggering the scan
             @return a boolean true if there are any WebBrowserSession still active
             @author  ddavidson
205       <!=========================================================================>*/

          protected boolean expireWebBrowserSessions( HttpServletRequest req )
          {
             HttpSession session = req.getSession( false );
210
             int numLeft = 0;
             WebLoginContext lc = null;
             if ( session != null )
             {
215             lc = (WebLoginContext)session.getValue( WebBrowserSession.WBS_KEY_LOGINCONTEXT );
             }

             if ( lc != null )
             {
220             String timeoutStr = (String)lc.get( lc.PROP_TIMEOUT );
                int to = 300000; // default to 5 minutes (in milliseconds)
                if ( timeoutStr != null )
                   to = Integer.parseInt(timeoutStr) * 1000;

225             // search over all session properties and look for any that
                // are our WebBrowserSession.
                String names[] = session.getValueNames();
                for ( int i=0; i<names.length; i++ )
                {
230                if ( names[i].startsWith( WebBrowserSession.WBS_KEY_BROWSERSESSION ) )
```

```
            {
               WebBrowserSession wbs = (WebBrowserSession)session.getValue( names[i] );
               if ( wbs.getLastAccessInterval() >= to )
               {
235               session.removeValue( names[i] );
               }
               else
               {
                  numLeft++;
240               }
            }
         }
      }
      return (numLeft > 0);
245   }

      /**<!===== getWebLoginContext ==========================================>
         Find the web login context, if one exists for the current HTTP session.
         @param  req The HttpServletRequest to search.
250      @return a WebLoginContext for the given session, or null if not found.
         @author  ddavidson
      <!=====================================================================>*/
      protected WebLoginContext getWebLoginContext( HttpServletRequest req )
      {
255      // get session, but don't create if not found
         HttpSession session = req.getSession( false );
         if ( session != null )
         {
            return (WebLoginContext)session.getValue(
260                                    WebBrowserSession.WBS_KEY_LOGINCONTEXT );
         }
         return null;
      }

265   /**<!===== createWebLoginContext =======================================>
         Create a WebLoginContext for the current session.  This WebLoginContext
         will be available to all servlets for this session.  It is placed
         under a "well known" key (Ie. "com.controlj") in the HttpSession.

270      @param req The request triggering the login
         @param sc The servlet's configuration (this should be the login servlet)
         @param session The HttpSession under which data is stored.
         @param security The core's CorbaLoginContext to login under
         @param user User name doing the login.
275      @return a WebLoginContext
```

```
      @author  ddavidson
    <!=============================================================================>*/

280 private WebLoginContext createWebLoginContext(
      HttpServletRequest req, ServletConfig sc, HttpSession session,
      Operator operator, String user )
    {
      // construct the actual WebLoginContext
285   WebLoginContext lc = new WebLoginContext( operator,
            session.getId(), user + "@" + req.getRemoteAddr() );

      // place it into the HttpSession object
      session.putValue( WebBrowserSession.WBS_KEY_LOGINCONTEXT, lc );
290
      // extract properties from the servet config and place into user context
      Enumeration enum = sc.getInitParameterNames();
      while ( enum.hasMoreElements() )
      {
295     // enumerate over all servlet's config props and put in the ones we know
        boolean found = false;
        String name = (String)enum.nextElement();
        String p = sc.getInitParameter( name );
        for ( int i=0; i<WebLoginContext.propList.length; i++ )
300     {
          if ( name.equalsIgnoreCase( WebLoginContext.propList[i] ) )
          {
            lc.put( WebLoginContext.propList[i], p );
            found = true;
305       }
        }
        if ( ! found )
        {
          log( "Property name [" + name + "] is unexpected" );
310       // FIX XXX do we care about the ones we don't recognize?
        }
      }

      // set the session specific timeout value if specified in the config
315   String p = (String)lc.get( lc.PROP_TIMEOUT );
      if ( p != null )
        session.setMaxInactiveInterval( Integer.parseInt(p) );

      return lc;
320 }

    /**<!===== addStdJavaScriptReturn ========================================>
```

```
      Insert anything in the "jsreturn" request parameter as literal JavaScript
      code to the end of the current list.
      @param  req   The HttpServletRequest object.
      @param  list  The JscriptMsgList response list.
      @author  ddavidson
   <!=========================================================================>*/

   protected void addStdJavaScriptReturn( HttpServletRequest req, JscriptMsgList list )
   {
      String js = req.getParameter( "jsreturn" );
      if ( js != null )
      {
         // add item as literal text to the end
         list.addItem( new JscriptSimple( js ) );
      }
   }

   /**<!===== stdHeader ========================================================>
      Output our standard TreeRequest HTML header for JavaScript response.
      @param  out  HTML output stream
      @param  wbs  The current WebBrowserSession
      @param  seq  The JavaScript supplied sequence number
      @author  ddavidson
   <!=========================================================================>*/

   protected void stdHeader( PrintWriter out, WebBrowserSession wbs, String seq )
   {
      out.println("<HTML>");
      out.println("<SCRIPT LANGUAGE=\"JavaScript\">");
      out.println("// Time served is " + new Date() );

      // if the user context is new, tell the java script what it's ID is
      if ( wbs != null && wbs.isNew() )
         out.println("window.parent.setWBS( " + wbs.getId() + " )");

      if ( seq != null )
      {
         out.println("if (window.parent.reqStarted( " + seq + " )){");
      }
   }

   /**<!===== stdFooter ========================================================>
      Output our standard HTML footer for JavaScript response.
      @param  out  HTML output stream
      @param  seq  The JavaScript supplied sequence number
      @param  ack  Our acknowledgement sequence number.
```

```
      @author  ddavidson
370   <!=============================================================================>*/

      protected void stdFooter( PrintWriter out, String seq, String ack )
      {
         if ( seq != null && ack != null)  // tree request
375      {
            out.println("window.parent.reqFinished( " + seq + ", " + ack + " )\n}");
            out.println("top.setNavBusy(false)");
         }
         out.println("</SCRIPT></HTML>");
380   }

      /**<!===== stdContent ================================================>
         Output standard page response.  Sets HTML content and cache control,
         calls stdHeader(), writes the comment, outputs the list, calls stdFooter()
385      and then closes the output stream.

         @param  req   The HttpServletRequest
         @param  resp  The HttpServletResponse
         @param  wbs   The current WebBrowserSession
390      @param  list  The JScriptMsgList of items to send
         @param  comment   And arbitrary javascript comment to append
         @author  ddavidson
      <!=============================================================================>*/

395   protected void stdContent( HttpServletRequest req, HttpServletResponse resp,
                                 WebBrowserSession wbs, JscriptMsgList list, String comment)
         throws IOException
      {
         resp.setContentType("text/html");
400      resp.setHeader( "Cache-Control", "no-cache" );

         PrintWriter out = new PrintWriter(resp.getOutputStream());
         String seq = req.getParameter( "seq" );

405      // insert any standard javascript footer code
         addStdJavaScriptReturn( req, list );

         // write standard header
         stdHeader( out, wbs, seq );
410
         // include any comment
         if ( comment != null )
            out.println( comment );
```

```
415       // write the body
          if ( list != null )
             list.write( out );

          // write standard footer
420
          stdFooter( out, seq, (list != null) ? list.getSequence() : "" );

          // and flush the output
          out.close();
425    }
       /**<!===== redirectTimeout ================================================>
          gracefully handles HTTPSession timeouts.
          @param  resp  HttpServletResponse
          @param  seq  The JavaScript supplied sequence number
430       @param  ack  Our acknowledgement sequence number.
          @author  ddavidson
       <!=========================================================================>*/
       protected void redirectTimeout (HttpServletResponse resp)
          throws IOException
435    {
          resp.setContentType("text/html");
          resp.setHeader( "Cache-Control", "no-cache" );
          PrintWriter out = new PrintWriter(resp.getOutputStream());

440       out.println("<HTML>");
          out.println("<SCRIPT LANGUAGE=\"JavaScript\">");
          out.println("window.top.location.href=\"/common/navpane/timeout.html\"");
          out.println("</SCRIPT></HTML>");
          out.close();
445    }

       /**<!===== getChangesFromCore ============================================>
          Gets the changes from the primitive binder and sends them down to
          javascript.
450
          @param PrimitiveBinder pb        - Primitive Binder (what else did you think).
          @param String getValuesFromField - Whether to get the values from the
                                             field ("true" or "false").
          @param JscriptMsgList list       - List to add the jscript code to.
455
          @author  jomisore
       <!=========================================================================>*/

       protected void getChangesFromCore(PrimitiveBinder pb,
460                                      String getValuesFromField,
```

```
                                        JscriptMsgList list)
        {
           synchronized ( pb )
           {
465           Vector changes;

              if ((getValuesFromField != null) && getValuesFromField.equals("true"))
              {
                  changes = pb.getPrimitiveValueChangesFromField();
470           }
              else
              {
                  changes = pb.getPrimitiveValueChanges();
              }
475
              Enumeration enum = changes.elements();

              while ( enum.hasMoreElements() )
              {
480              PrimitiveBinder.PrimitiveValueChange change =
                    (PrimitiveBinder.PrimitiveValueChange)enum.nextElement();
                 list.addItem( new JscriptSetMasterValue( change.id, change.value ) );
              }
           }
485     }

        ////////////////////////////////////////////////////////////////////////////
        ////////////////////////////// Standard debug //////////////////////////////
        ////////////////////////////////////////////////////////////////////////////
490
        static private void debugInfo(String message)  { com.controlj.green.common.Debug.info(
"ServletBase",message); };
        static private void debugError(String message) {
com.controlj.green.common.Debug.error("ServletBase",message); };

     }
```

Exhibit 13

```
/*============================================================================
                    AUTOMATED LOGIC CORPORATION
             Copyright (c) 1999 - 2000 All Rights Reserved
      This document contains confidential/proprietary information.
============================================================================

   @(#)JscriptMsgList.java

   Author(s) ddavidson

   $Log: /green/SOURCE/COM/controlj/green/web/servlets/util/JscriptMsgList.java $
 *
 * 2     8/12/99 10:15a Ddavidson
 * Rewrite using new core for Milestone 2
 *
 * 1     6/30/99 9:27a Ddavidson

============================================================================*/

package com.controlj.green.web.servlets.util;

import java.util.*;
import java.io.PrintWriter;

/**<!============================================================================>
   Contains a list of JscriptMsg objects and manages insertion order
   and output of the entire list.  Create items, add them, and write them.
   @author ddavidson.
<!============================================================================>*/

public class JscriptMsgList
{
   /**<!----- seq ------------------------------------------------------------>
      Current request sequence id we are sending items for.
   <!------------------------------------------------------------------------>*/
   private String seq;

   /**<!----- items ---------------------------------------------------------->
      List of JscriptMsgs.
   <!------------------------------------------------------------------------>*/
   private Vector items;

   /**<!===== JscriptMsgList ================================================>
      Create empty list with the associated sequence value.
      @param  seq
      @author  ddavidson
   <!============================================================================>*/

   public JscriptMsgList( String seq )
   {
```

```
      this.items = new Vector();
      this.seq = seq;
   }

/**<!===== JscriptMsgList ================================================>
   Create empty list with an empty sequence value for items that don't
   request future acknowledgement.
   @author  ddavidson
<!=======================================================================>*/

   public JscriptMsgList()
   {
      this( "" );
   }

/**<!===== getSequence ====================================================>
   Return the request sequence value this list is associated with.
   @return a String
   @author  ddavidson
<!=======================================================================>*/

   public String getSequence()
   {
      return seq;
   }

/**<!===== addItem ========================================================>
   Adds item to list based on requested order by the item.
   Set's the items sequence value to be that of this containing list.

   @param  item
   @author  ddavidson
<!=======================================================================>*/

   public void addItem( JscriptMsg item )
   {
      // if item has an insertion order the determine where to insert it
      if ( item.hasInsertOrder() )
      {
         for ( int i=0; i<items.size(); i++ )
         {
            if ( item.insertBefore( items.elementAt(i).getClass() ) )
            {
               items.insertElementAt( item, i );
               item = null; // don't fall thru to tail insertion
               break;
            }
         }
      }
      // If still around, then add it to the end.
```

```
            if ( item != null )
            {
               items.addElement( item );
            }
105      }

         /**<!===== write ============================================================>
            Invokes "encodeHtml( PrintWriter )" on each item in order.
            @param  out   The PrintWriter to output the items to.
110         @author  ddavidson
         <!============================================================================>*/

         public void write( PrintWriter out )
         {
115         Enumeration enum = items.elements();
            while ( enum.hasMoreElements() )
            {
               JscriptMsg j = (JscriptMsg)enum.nextElement();
               j.encodeHtml( out );
120         }
         }
      }
```

Exhibit 14

```
/*============================================================================
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
      This document contains confidential/proprietary information.
============================================================================

    @(#)JscriptSimple.java

    Author(s) ddavidson

    $Log: /green/SOURCE/COM/controlj/green/web/servlets/util/JscriptSimple.java $
 *
 * 5     12/02/99 3:34p Ddavidson
 * added add(boolean) method
 *
 * 4     11/09/99 8:28a Ddavidson
 * Added use of CJIO.js method
 *
 * 3     10/07/99 4:15p Ddavidson
 * misc routines to help with scheduling JSP main page
 *
 * 2     8/12/99 10:15a Ddavidson
 * Rewrite using new core for Milestone 2
 *
 * 1     6/30/99 9:27a Ddavidson

============================================================================*/

package com.controlj.green.web.servlets.util;

import java.io.PrintWriter;
import com.controlj.green.common.CJIO;


/**<!============================================================================>
     An interface for simple JscriptMsg items.  It handles generic text
     as well a properly formatting a JavaScript method.<br>

     <CODE>
           startMethod( "new window.parent.Icon" );<br>
           addQuotedArg( name );<br>
           addArg( index );<br>
           addNewline( "  " ); <br>
           addQuotedArg( imageURL );<br>
           endMethod( true );<br>
    </CODE>
```

```
      Would Generate:<br>
      <CODE>
         new window.parent.Icon( "somename", 1,<br>
50         "/icons/subdir/myfile.gif" );<br> ♀          </CODE>
      @author ddavidson.
   <!=========================================================================>*/

55 public class JscriptSimple implements JscriptMsg
   {
      /**<!----- jscript ------------------------------------------------------>
         The JavaScript text to output.
      <!------------------------------------------------------------------------>*/
60    private StringBuffer jscript;

      /**<!----- isFirstArg --------------------------------------------------->
         Are we at the first arg on a line?
      <!------------------------------------------------------------------------>*/
65    private boolean isFirstArg = false;

      /**<!===== JscriptSimple ================================================>
         Create with specified initial text.
         @param  jscript The init string of JavaScript to write.
70       @author  ddavidson
      <!=========================================================================>*/

      public JscriptSimple( String jscript )
      {
75       this.jscript = new StringBuffer( jscript );
      }

      /**<!===== JscriptSimple ================================================>
         Create with empty text.
80       @author  ddavidson
      <!=========================================================================>*/

      public JscriptSimple()
      {
85       this.jscript = new StringBuffer();
      }

      /**<!===== encodeHtml ===================================================>
         Method to output HTML text to self describe instance to the
90       associated JavaScript.
         @param  out  The PrintWriter object associated with the output stream
         @author  ddavidson
      <!=========================================================================>*/
```

```
95    public void encodeHtml( PrintWriter out )
      {
         out.println( jscript.toString() );
      }

100   /**<!===== getText ============================================================>
         Return the current text.
         @return a String
         @author  ddavidson
      <!============================================================================>*/

105   public String getText()
      {
         return jscript.toString();
      }

110   /**<!===== hasInsertOrder =====================================================>
         Default doesn't care - add to end of list. Override in subclass.
         @return false
         @author  ddavidson
115   <!============================================================================>*/

      public boolean hasInsertOrder()
      {
         return false;
120   }

      /**<!===== insertBefore ========================================================>
         Doesn't care - always return false.  Override in subclass.
         @param  testClass   The test class
125      @return a boolean   Always false.
         @author  ddavidson
      <!============================================================================>*/

      public boolean insertBefore( Class testClass )
130   {
         return false;
      }

      /**<!===== add =================================================================>
135      Add the literal string to the text with no formatting.
         @param  s  The string to add.
         @author  ddavidson
      <!============================================================================>*/
```

```
140    public void add( String s )
       {
          jscript.append(s);
       }

145    /**<!===== addQuotedArg ================================================>
          Add the string as a quote argument, handling commas where necessary.
          @param  s  The string to add.
          @author  ddavidson
       <!=======================================================================>*/
150
       public void addQuotedArg( String s )
       {
          if (isFirstArg )
          {
155          isFirstArg = false;
          }
          else
          {
             jscript.append( ", " );
160       }
          jscript.append( '"' );
          jscript.append( CJIO.js(s) );
          jscript.append( '"' );
       }
165
       /**<!===== addQuotedArg ================================================>
          Add the int as a quote argument, handling commas where necessary.
          @param  i The integer to add.
          @author  ddavidson
170    <!=======================================================================>*/

       public void addQuotedArg( int i )
       {
          addQuotedArg( Integer.toString(i) );
175    }

       /**<!===== addQuotedArg ================================================>
          Add the long as a quote argument, handling commas where necessary.
          @param  l The long to add.
180       @author  ddavidson
       <!=======================================================================>*/

       public void addQuotedArg( long l )
       {
185       addQuotedArg( Long.toString(l) );
```

```
    }

    /**<!===== addArg ============================================================>
       Add the string as an unquoted argument, handling commas where necessary.
190    @param  s The String to add.
       @author  ddavidson
    <!============================================================================>*/

    public void addArg( String s )
195 {
       // add a comma before all but the first arg on a line
       if (isFirstArg )
       {
          isFirstArg = false;
200    }
       else
       {
          jscript.append( ", " );
       }
205    jscript.append( s );
    }

    /**<!===== addArg ============================================================>
       Add the int as an argument, handling commas where necessary.
210    @param  i  The integer to add.
       @author  ddavidson
    <!============================================================================>*/

    public void addArg( int i )
215 {
       addArg( Integer.toString(i) );
    }

    /**<!===== addArg ============================================================>
220    Add the long as an argument, handling commas where necessary.
       @param  l  The long to add.
       @author  ddavidson
    <!============================================================================>*/

225 public void addArg( long l )
    {
       addArg( Long.toString(l) );
    }

230 /**<!===== addArg ============================================================>
       Add the boolean as an argument, handling commas where necessary.
```

```
      @param  b  The boolean to add.
      @author  ddavidson
   <!=========================================================================>*/
235
   public void addArg( boolean b )
   {
      addArg( b ? "true" : "false" );
   }
240

   /**<!===== addNewline ========================================================>
      Add a newline to the current text, handling commas where necessary.
      @author  ddavidson
245<!=========================================================================>*/

   public void addNewline()
   {
      // If no argument on this line yet, just add the newline,
250   // else append a command and start a new line
      if ( isFirstArg )
      {
         jscript.append( '\n' );
      }
255   else
      {
         jscript.append( ",\n" );
         isFirstArg = true;
      }
260}

   /**<!===== addNewline ========================================================>
      Same as addNewline() but inserts the prefix text on the next line.
      This is useful for indenting subsequent lines.
265
      @param  nextPrefix
      @author  ddavidson
   <!=========================================================================>*/

270public void addNewline( String nextPrefix )
   {
      addNewline();
      jscript.append( nextPrefix );
   }
275
   /**<!===== startMethod =======================================================>
      Starts a method declaration with the specified name.
```

```
      Appends a "(" and after the specified name.  The name may include
      any text, for example "foo = new func".<br>
280
      <b>WARNING: Does not support nested methods currently.</b>.

      @param  name
      @author  ddavidson
285   <!=========================================================================>*/

      public void startMethod( String name )
      {
         jscript.append( name );
290      jscript.append( "( " );
         isFirstArg = true;
      }

      /**<!===== endMethod ========================================================>
295      Ends a method declaration, optional adding a semicolon after ")".
         If ending a nested method, speciffy false.
         @param  addSemicolon  True if a semicolon should be added after the
            function close.
         @author  ddavidson
300   <!=========================================================================>*/

      public void endMethod( boolean addSemicolon )
      {
         if ( addSemicolon )
305      {
            jscript.append( " );\n" );
         }
         else
         {
310         jscript.append( " )" );
         }
         isFirstArg = true;
      }

315   /**<!===== beginList ========================================================>
         Mark the current state at the begining of a comma separated list.
         @return a String
         @author  ddavidson
      <!=========================================================================>*/
320
      public void beginList()
      {
         isFirstArg = true;
```

```
        }
325
        /**<!===== toString ======================================================>
           Return the  HTML output as a string.
           @return a String
           @author  ddavidson
330     <!=====================================================================>*/

        public String toString(  )
        {
           return jscript.toString();
335     }

    }
```

Exhibit 15

```
/*============================================================================
                    AUTOMATED LOGIC CORPORATION
            Copyright (c) 1999 - 2000 All Rights Reserved
      This document contains confidential/proprietary information.
============================================================================

   @(#)JscriptSetMasterValue.java

   Author(s) ddavidson

   $Log: /green/SOURCE/COM/controlj/green/web/servlets/util/JscriptSetMasterValue.java $
 *
 * 2     8/26/99 10:46a Ddavidson
 * Added Submit/Update servlet for primitives.
 * Moved stdHeader/Footer/Content methods into ServletBase
 *
 * 1     8/26/99 8:26a Ddavidson

============================================================================*/

package com.controlj.green.web.servlets.util;

import java.io.*;

/**<!========================================================================>
   A JScriptSimple derived class to send primitive value change messages.
   They insert at the tail of a list.
   @author ddavidson.
<!========================================================================>*/

public class JscriptSetMasterValue extends JscriptSimple
{

   /**<!===== JscriptSetMasterValue ==========================================>
      Inserts a PrimitiveBase.setPrimitiveMasterValue for the given primitive
      ID and value.
      @param  id     The primitive's ID.
      @param  value  The primitive's new value.
      @author  ddavidson
   <!========================================================================>*/

   public JscriptSetMasterValue( String id, String value )
   {
      super( "" );
      startMethod( "window.parent.PrimitiveBase.setPrimitiveMasterValue" );
      addQuotedArg( id );
      addQuotedArg( value );
      endMethod( true );
   }
```